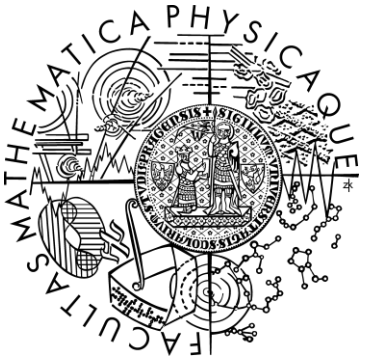
Univerzita Karlova v Praze

Matematicko-fyzikální fakulta

DIPLOMOVÁ PRÁCE



Michal Drobný

Metody řešení vybraných dopravních problémů a jejich implementace

Katedra aplikované matematiky

Vedoucí diplomové práce: doc. RNDr. Libuše Grygarová, DrSc.

Studijní program: Informatika

Studijní obor: Softwarové systémy

Praha 2013

Poděkování

Rád bych poděkoval všem, kteří mi byli jakkoli nápomocni při psaní této práce. Děkuji vedoucí mé diplomové práce docentce Libuši Grygarové za odborné konzultace a rady. Dále děkuji celé své rodině, zvláště pak mé matce Blance Drobné a blízkým za jejich trpělivost a všestrannou podporu. Zvláštní poděkování patří bratru Martinu Drobnému za cenné rady při zpracování výsledků, Lence Kubíkové za revize textu a doc. Petru Kolmanovi za zapůjčení literatury.

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně a výhradně s použitím citovaných pramenů, literatury a dalších odborných zdrojů.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona v platném znění, zejména skutečnost, že Univerzita Karlova v Praze má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona.

V ..... dne .....

Název práce: Metody řešení vybraných dopravních problémů a jejich implementace

Autor: Bc. Michal Drobný

Katedra: Katedra aplikované matematiky

Vedoucí práce: doc. RNDr. Libuše Grygarová DrSc., Katedra aplikované matematiky

Abstrakt: S různými typy dopravních problémů se v praxi setkáváme velmi často. Tento problém lze chápat především jako rozvoz zboží od dodavatelů k odběratelům s cílem minimalizace distribučních nákladů. Reálné dopravní problémy se od těch obecných liší především uvažovanými restrikcemi, což mohou být například kapacity vozidel a objednávek, časová okna a různá další speciální distribuční omezení. Problematiku dopravního problému formuloval již F. L. Hitchcock v roce 1941 a od té doby bylo popsáno mnoho stochastických a nedeterministických metod pro řešení dopravního problému, nicméně při zavedení distribučních restrikcí pro řešení reálných problémů jsou tyto metody obtížně aplikovatelné.

Tato práce poskytuje kompilaci nejznámějších deterministických metod vhodných pro řešení dopravních problémů, přičemž metody vhodné pro řešení reálných dopravních problémů jsou popsány podrobněji. Postup řešení pro vybrané metody je demonstrován na jednoduchých příkladech a výsledky porovnány s výsledky řešení ostatních metod. Na základě analýzy těchto metod jsou navrženy nové metody pro řešení reálných dopravních problémů, které jsou implementovány a jejich výsledky porovnány s metodami, které poskytuje komerční softwarový produkt.

Klíčová slova: Dopravní problém, úloha obchodního cestujícího, přiřazovací problém, víceokruhový okružní problém.

Title: Methods for solving selected vehicle routing problems and their implementations

Author: Bc. Michal Drobný

Department: Department of Applied Mathematics

Supervisor: doc. RNDr. Libuše Grygarová DrSc., Department of Applied Mathematics

Abstract: Various types of transportation issues are a common practice. The issue may be approached mainly as the distribution of products from suppliers to consumers while minimising distribution costs. The difference of real transportation issues predominantly relates to the considered restrictions, such as capacities of vehicles and orders, time windows and other special distribution restrictions. Transportation issues were already defined by F.L. Hitchcock in 1941 and since then, a wide range of stochastic and non-determinist methods providing solutions to transportation issues have been developed. Nevertheless, introducing distribution restrictions in resolving real-life problems makes it difficult for such methods to be applied.

This thesis provides a compilation of the well-known determinist methods that may be used to resolve transportation issues. The methods that are appropriate for resolving real issues are discussed in more detail. The solution procedure of the selected method is demonstrated using simple examples and the results are compared with the results of other methods. An analysis of the above methods is used to design and implement new methods to resolve real transportation issues, their results being compared with the methods provided by the commercial software product.

Keywords: Transportation problem, travelling salesman problem, bin packing problem, vehicle routing problem.

Obsah

Úvod

V praxi se velmi často setkáváme s různými typy dopravních problémů. Tyto problémy lze chápat především jako rozvoz zboží od dodavatelů k odběratelům s cílem minimalizace distribučních nákladů. Navíc při řešení reálných dopravních problémů jsou velmi často zavedeny další restrikce (jako kapacita vozidel a objednávek, časová okna, maximální provozní doba vozidel a další speciální distribuční podmínky), které omezují množinu přípustných řešení problémů.

Problematiku klasického dopravního problému formuloval již F. L. Hitchcock v roce 1941, kdy navrhnul i jednoduchou metodu jeho řešení [1], nicméně podrobněji byla tato zkoumána až G. Dantzigem, který ve své publikaci *Application of the Simplex Method to a Transportation Problem* [2] popsal upravenou simplexovou metodu a definoval tak jednu z nejpoužívanějších metod pro řešení klasického dopravního problému.

V současné době je popsáno mnoho stochastických a nedeterministických metod pro řešení klasického dopravního problému, nicméně při zavedení distribučních restrikcí pro řešení reálných problémů jsou tyto metody obtížně aplikovatelné [3]. Také metody klastrování a prořezávání se zdají být nevhodné z důvodu vysoké výpočetní složitosti.

V praxi se s klasickým dopravním problémem můžeme setkat málokdy. Tato práce se zabývá kompilací nejznámějších deterministických metod pro řešení reálných víceokruhových dopravních problémů a návrhem nových metod pro řešení těchto problémů. V rámci návrhu metod nebyly uvažovány podmínky časových oken, přestože se s nimi v praxi setkáváme velmi často. Problematika časových oken je velice rozsáhlá a přesahuje rozsah této práce.

První kapitola je věnována popisu speciálních problémů, se kterými se lze setkat v různých fázích dekompozice reálného dopravního problému. Pro každý problém uvedeme matematický model, kterým se dále budeme zabývat v následujících kapitolách.

Druhá kapitola podrobně zpracovává nejznámější metody řešení těchto problémů, přičemž při zpracování těchto metod byly opraveny určité nepřesnosti a metody byly

popsány detailněji. Postup řešení pro vybrané metody je demonstrován na jednoduchých příkladech a výsledky řešení byly vzájemně porovnány.

Další kapitola se věnuje popisu reálného dopravního problému a návrhu metod pro řešení jeho vybraných variant. V rámci návrhu nových metod dekomponujeme problém řešení reálného dopravního problému na jednotlivé fáze. Pro tyto fáze navrhujeme nové metody na základě vybraných algoritmů pro řešení speciálních problémů popsaných v první kapitole.

V poslední kapitole aplikujeme postupy navržených metod na vybrané reálné problémy. Jedná se o problém rozvozu lístků pro sčítání lidu z Prahy do krajských měst a problém distribuce zboží pro větší množství objednávek z centrálního skladu v Prostějově do různých míst v České republice. Výsledky řešení problému rozvozu lístků sčítání lidu jsou pak porovnány s výsledky řešení pomocí vybraných metod popsaných v rámci druhé kapitoly. Výsledky řešení problému distribuce potravin porovnáváme s výsledky získanými vybranými komerčními algoritmy pro řešení maloobchodních i velkoobchodních reálných dopravních problémů.

1 Speciální problémy

V následujících paragrafech se zaměříme na speciální problémy, které můžeme využít v souvislosti s konstrukcí metod pro řešení námi uvažovaných vybraných dopravních problémů. Jedná se především o známé optimalizační problémy typu *„Úloha obchodního cestujícího“, „Přiřazovací problém“* a *„Dopravní problém“*.

V dalších částech kapitoly uvedeme jejich matematické modely a dále konkrétní algoritmy pro jejich řešení.

1.1 Úloha obchodního cestujícího

Úloha obchodního cestujícího je také často označována jako *Okružní dopravní problém* nebo *Problém listonoše*, nicméně dle referencí zahraniční literatury se dále budeme držet označení *Úloha obchodního cestujícího* (Travelling Salesman Problem).

Jedná se o obtížný diskrétní optimalizační problém, který byl popsán již v první polovině 19. století významnými matematiky W. R. Hamiltonem a T. Kirkmanem. Podrobněji se jím zabýval až Karl Menger na začátku 20. století a brzy nato Hassler Whitney z Princeton University, který ho ve svých publikacích definoval jako Travelling Salesman Problem[4]. Později byl důkladně prostudován a jeho NP-úplnost dokázal Karp [5].

Tuto úlohu lze popsat jako hledání cyklu v ohodnoceném grafu tak, aby cyklus obsahoval stanovené vrcholy a zároveň měl co nejnižší celkové ohodnocení. Problém lze tedy chápat jako snahu cestujícího navštívit všechny požadované destinace s cílem najet co nejméně kilometrů a vrátit se do výchozího místa.

Formálněji lze tento problém definovat takto:

Mějme ohodnocený graf $G(V, E)$ s nezáporným ohodnocením $H: E \rightarrow \mathbb{R}_0^+$, dále mějme množinu vrcholů $M = \{v | v \in V\}$. Úlohou je najít posloupnost vrcholů $A = \{v_{j_1}, v_{j_2}, \dots, v_{j_k}\}$ tak, aby hrany $h \in E$ mezi hledanými vrcholy v_{j_i} tvořily v grafu G cyklus. Dále aby

posloupnost A obsahovala každý vrchol z množiny M právě jednou a zároveň aby celkové ohodnocení hran mezi hledanými vrcholy bylo minimální. Jde tedy o úlohu:

$$\forall(w \in M)\ \exists(i \in \{1, \dots, k\}): \left(w = v_{j_i}\right), \tag{R 1.1}$$

$$\forall(i \in \{1,2, \dots, k\}): \left(\left(v_{j_i}; v_{j_{(i+1) mod\, k}}\right) \in E\right), \tag{R 1.2}$$

$$\sum_{i=1}^{k} h\left(\left(v_{j_i}; v_{j_{(i+1) mod\, k}}\right)\right) \xrightarrow[min]{}, \tag{R 1.3}$$

kde $h\left(\left(v_i; v_j\right)\right) \in \mathbb{R}_0^+$ je ohodnocení hrany $\left(v_i; v_j\right) \in E$, pro $v_i, v_j \in V$.

Z hlediska složitosti je úloha obchodního cestujícího NP-úplný problém. Pro řešení tohoto problému byla navržena celá řada heuristických algoritmů, kterými se dále budeme zabývat v paragrafu 2.1. Existují ale i algoritmy, které naleznou optimální řešení. Jedná se například o algoritmus dynamického programování, který je popsán v paragrafu 2.1.1.1.

1.2 Přiřazovací problém

Přiřazovací problém, častěji označován jako „bin-packing problem“, je diskrétní optimalizační problém, jehož cílem je minimalizace kontejnerů (příp. nádob), do kterých se vejdou kapacitně rozdílné objekty ze zadané vstupní množiny.

V literatuře je popsáno mnoho modifikací přiřazovacího problému, mezi nejčastější modifikaci řadíme požadavek stejné kapacity všech kontejnerů (nádob). Nicméně my se dále budeme držet obecného modelu (popsaného níže), neboť odpovídá problému přiřazení objednávek k jednotlivým vozidlům v dopravním problému, který je pro nás prioritní.

Přiřazovací problém lze jednoduše popsat jako přiřazení kapacitně rozdílných objektů ke kontejnerům, které mohou mít taktéž nestejné kapacity. Cílem úlohy je minimalizace počtu kontejnerů.

Z hlediska složitosti je přiřazovací problém NP-úplný, což dokázal již Schlag, Liao a Wong [6]. Z tohoto důvodu se při řešení zaměříme na heuristické algoritmy, pro které ukážeme, že

pracují efektivně a jejich řešení je pro naše účely dostačující. Tyto poznatky později využijeme při návrhu konkrétních algoritmů pro vybrané dopravní problémy v kapitole 3.

1.3 Dopravní problém

Dopravní problém, někdy nazýván jako distribuční úloha, je diskrétní optimalizační problém, jehož cílem je distribuce zboží mezi dodavateli a odběrateli, přičemž požadujeme minimalizaci nákladů při přepravě zboží.

Poprvé byl formulován F. L. Hitchcockem v roce 1941, který navrhnul i jednoduchou metodu řešení [1], nicméně podrobněji byl zkoumán až G. Dantzigem, který ve své publikaci *Application of the Simplex Method to a Transportation Problem* [2] popsal upravenou simplexovou metodu a definoval tak jednu z nejpoužívanějších metod pro řešení klasického dopravního problému.

Označme $k \in \mathbb{N}$ počet dodavatelů a $l \in \mathbb{N}$ počet odběratelů. Dopravní problém lze chápat jako nalezení optimálního množství zboží, které má být převezeno od i-tého dodavatele k j-tému odběrateli, kde $i \in \{1,2, \dots, k\}$ a $j \in \{1,2, \dots, l\}$. Stěžejním požadavkem je minimalizace distribučních nákladů. Většinou také požadujeme uspokojení poptávky odběratelů i nabídky dodavatelů.

V současné době vzniká řada studií založených na heuristických algoritmech, které jsou efektivní z hlediska časové a implementační náročnosti [7] [8]. Vybrané metody jsou popsány v paragrafu 2.2.

Zde v rámci matematického modelu uvádíme obecnější variantu, která nepožaduje uspokojení nabídky dodavatelů. Tato varianta odpovídá požadavkům vybraných dopravních problémů, které budeme studovat v kapitole 3.

Dopravní problém je z hlediska složitosti NP-úplný a proto se při popisu metod jeho řešení zaměříme především na heuristické algoritmy, jejichž vybrané varianty podrobně popíšeme a dále využijeme v kapitole 3.

1.4 Matematické modely

Matematickým modelem rozumíme abstraktní model používající matematický zápis pro vyjádření reálného problému. Tyto modely převádí složité lingvistické vyjádření podmínek a omezení na strojově zpracovatelný matematický jazyk, který lze dále efektivně integrovat do obecných algoritmů.

1.4.1 *Matematický model úlohy obchodního cestujícího*

Pro úlohu obchodního cestujícího lze sestavit jednoduchý matematický model. Standardní úlohu obchodního cestujícího popsanou v paragrafu 1.1 rozšíříme dále o vícenásobné cykly a podmínku omezení počtu navštívených destinací pro každý cyklus. Toto rozšíření je vhodné pro zobecnění celého problému a následnou integraci na reálné dopravní problémy, kterými se budeme zabývat v dalších částech této práce.

Nejprve zavedeme vstupní proměnné, což bude především vyjádření nákladového ohodnocení jednotlivých tras mezi destinacemi a dále předdefinovaný počet cyklů společně s maximálním počtem navštívených destinací pro každý cyklus. Tyto předdefinované podmínky konkretizují reálný dopravní problém tak, aby mohl být jednoduše matematicky popsán.

Dále budeme v modelu využívat pomocné proměnné, které budou charakterizovat pořadí, v jakém jsou navštíveny uvažované destinace v nalezeném řešení celé úlohy obchodního cestujícího.

Konečně sestavíme matematické rovnice tak, aby všechny podmínky v rozšířené úloze obchodního cestujícího byly splněny.

Výchozí destinaci (chceme-li základnu) budeme indexovat 0 a nebudeme ji zahrnovat do destinací, které chceme navštívit.

Proměnné lze definovat takto:

$N \in \mathbb{N}$	..	Počet destinací, které chceme navštívit.
$P \in \mathbb{N}$	..	Maximální počet destinací, které lze navštívit v jednom cyklu.
$T \in \mathbb{N}, TP \geq N$	..	Počet cyklů.
$C_{ij} \in \mathbb{R}_0^+$ $i, j \in \{0,1, \dots, N\}, C_{ij} = 0\ pro\ i = j$	..	Nákladové ohodnocení trasy mezi i-tou a j-tou destinací.
$X_{ij} \in \{0,1\}$ $i, j \in \{0,1, \dots, N\}, X_{ij} = 0\ pro\ i = j$	..	Pomocná proměnná, která vyjadřuje, zda mezi i-tou a j-tou destinací povede trasa v daném řešení úlohy. Povede-li, pole $X_{ij} = 1$, v opačném případě $X_{ij} = 0$.
$\delta_{iz} \in \{1,2, \dots, P\}$, $z \in \{1,2, \dots T\}, i \in \{1,2, \dots, N\}$, $\delta_{iz} \neq \delta_{jz}\ pro\ i \neq j$	..	Pomocná proměnná vyjadřující pořadí, v jakém je navštívena i-tá destinace v rámci z-tého okruhu.

Dále sestavíme rovnice, které zajistí splnění podmínek úlohy obchodního cestujícího. Je zřejmé, že každá destinace musí být navštívena právě jednou, a proto i z každé destinace vede pouze jedna trasa. Pokud povolíme více cyklů, pak tuto podmínku nesplňuje pouze výchozí destinace (základna), pro kterou zavedeme podmínky zvlášť.

$$\sum_{i=1}^{N} X_{ij} = 1, \qquad \text{(R 1.4)}$$

$$\sum_{j=1}^{N} X_{ij} = 1, \qquad \text{(R 1.5)}$$

kde $i \in \{1,2, \dots, N\}$ a $j \in \{1,2, \dots, N\}$.

Pro výchozí destinaci (základnu) musí platit, že počet cest ze základny i do základny musí být roven počtu cyklů, tj. T.

$$\sum_{j=1}^{N} X_{0j} = T, \tag{R 1.6}$$

$$\sum_{i=1}^{N} X_{i0} = T. \tag{R 1.7}$$

Aby vznikly cykly, (kde základna bude vždy startovní i cílová destinace) je nutné zahrnout podmínku cyklu:

$$\begin{gathered} \delta_{iz} - \delta_{jz} + PX_{ij} \leq P - 1, \\ i, j \in \{1,2, \dots, N\}, i \neq j, z \in T. \end{gathered} \tag{R 1.8}$$

Je zřejmé, že podmínka nabývá na účinnosti pouze pokud $X_{ij} > 0$. Jinak je podmínka splněna vždy. Pokud $X_{ij} > 0$, pak nutně i $\delta_{jz} > \delta_{iz}$. V jednom cyklu je vždy $\delta_{iz}, \delta_{jz} \in \{1, 2, \dots, P\}$. Odvození této podmínky ukázal Miller [7].

Za těchto podmínek se snažíme o minimalizaci celkových nákladů na dopravu

$$\sum_{i=0}^{N} \sum_{j=0}^{N} C_{ij} X_{ij} \xrightarrow{min}. \tag{R 1.9}$$

Jak bylo zmíněno, úloha obchodního cestujícího je obecně z hlediska obtížnosti NP-úplná, tudíž se pro její řešení používají různé heuristické optimalizace. Tyto metody sice nezaručují optimální řešení, zato však v praxi poskytují uspokojivé výsledky.

1.4.2 *Matematický model přiřazovacího problému*

Mějme množinu $M = \{\mathfrak{y}_1 u_1, \mathfrak{y}_2 u_2, \dots, \mathfrak{y}_m u_m\}$, kde $m \in \mathbb{N}$ je počet kapacitně rozdílných objektů, $u_i \in \mathbb{R}^+$ je kapacita i-tého objektu a $\mathfrak{y}_i \in \mathbb{N}$ je jeho kardinalita, $i \in \{1,2, \dots, m\}$. K dispozici máme $l \in \mathbb{N}$ kontejnerů. Kapacitu j-tého kontejneru značíme $v_j \in \mathbb{R}$, kde $j \in \{1,2, \dots, l\}$. Bez újmy na obecnosti předpokládejme, že u_1 je nejnižší kapacita.

Pro každý j-tý kontejner zavedeme vektor $T_j = (t_{j1}, t_{j2}, \dots, t_{jm})$, kde $t_{ji} \in \mathbb{Z}_0^+$ je počet objektů typu u_i obsažených v j-tém kontejneru, $i = \{1,2, \dots, m\}, j = \{1,2, \dots, l\}$.

Řekneme, že j-tý kontejner je nasycen, pokud platí následující dvě podmínky:

$$\sum_{i=1}^{m} t_{ji} u_i \leq v_j, \tag{R 1.10}$$

$$u_1 + \sum_{i=1}^{m} t_{ji}\, u_i > v_j. \tag{R 1.11}$$

Úlohou přiřazovacího problému je najít přiřazení $T = \{T_1, T_2, \dots, T_l\}$ tak, aby byl každý objekt z množiny $\{1, 2, \dots, m\}$ přiřazen právě tolikrát, kolik čítá jeho kardinalita, a rovněž aby počet kontejnerů l byl minimální:

$$\sum_{j=1}^{l} t_{ji} = \mathfrak{y}_i, \qquad i = \{1,2, \dots, m\}, \tag{R 1.12}$$

$$l \underset{min}{\longrightarrow}. \tag{R 1.13}$$

Jak bylo zmíněno, přiřazovací problém je obecně z hlediska obtížnosti NP-úplný, tudíž se pro jeho řešení používají různé heuristické optimalizace. Tyto metody sice nezaručují optimální řešení, zato však v praxi poskytují uspokojivé výsledky.

1.4.3 *Matematický model dopravního problému*

Mějme seznam nabídek k dodavatelů $D = \{D_1, D_2, \dots, D_k\}$, kde $D_i \in \mathbb{R}^+$, $i \in \{1,2, \dots, k\}$, značí nabídku i-tého dodavatele. Dále seznam poptávek l odběratelů $O = \{O_1, O_2, \dots, O_l\}$, kde $O_j \in \mathbb{R}^+$, $j \in \{1,2, \dots, l\}$, značí poptávku j-tého odběratele. Dále mějme matici $C_{ij} \in \mathbb{R}_0^+$, $i \in \{1,2, \dots, k\}$, $j \in \{1,2, \dots, l\}$, která charakterizuje nákladovou vzdálenost mezi i-tým dodavatelem a j-tým odběratelem. Nákladovou vzdáleností chápeme náklady na přesun jedné jednotky zboží mezi daným dodavatelem a odběratelem.

Tento model dopravního problému definuje nákladovou vzdálenost, nicméně v reálné praxi počítáme spíše s časovými náklady, případně s náklady přesunu celé kapacity vozidla mezi danými destinacemi. V modelu se také předpokládá, že distribuční náklady rostou lineárně s množstvím přepravovaného zboží. Tento předpoklad obvykle není reálný, nicméně zjednodušuje dopravní problém tak, abychom mohli jednoduše formulovat jeho model.

V obecné formulaci dopravního problému požadujeme, aby byla uspokojena alespoň poptávka odběratelů. Musí platit, že nabídka dodavatelů je rovna nebo převyšuje poptávku odběratelů

$$\sum_{i=1}^{k} D_i \geq \sum_{j=1}^{l} O_j. \tag{R 1.14}$$

Pokud tato podmínka není splněna, můžeme zavést fiktivního dodavatele $k+1$ s příslušným $C_{(k+1)j} \to \infty$, $j \in \{1,2,\dots,l\}$.

Dále zavedeme pomocné proměnné $X_{ij} \in \mathbb{R}_0^+$, $i \in \{1,2,\dots,k\}$, $j \in \{1,2,\dots,l\}$, které vyjadřují, kolik zboží bude převezeno mezi i-tým dodavatelem a j-tým odběratelem. Uvažujeme $X_{ij} \in \mathbb{R}_0^+$, nicméně obvykle nejsou jednotky zboží dělitelné a proto v reálném modelu se lze omezit na $X_{ij} \in \mathbb{N}$.

Pro každého j-tého odběratele musí platit, že jeho poptávka je uspokojena

$$\sum_{i=1}^{k} X_{ij} = O_j, \quad j \in \{1,2,\dots,l\}. \tag{R 1.15}$$

Cílem úlohy je minimalizace distribučních nákladů, tj.

$$\sum_{i=1}^{k}\sum_{j=1}^{l} X_{ij} C_{ij} \xrightarrow{min}. \tag{R 1.16}$$

2 Přehled známých metod

V následujících paragrafech se zaměříme na popis známých metod pro speciální problémy, které byly popsány v kapitole 1. Metody rozdělíme dle uvažovaných speciálních problémů.

2.1 Metody řešení úlohy obchodního cestujícího

Jak již bylo zmíněno, úloha obchodního cestujícího je NP-úplný problém, tudíž pro její řešení v podstatě neexistuje složitostně efektivní algoritmus, který by dával optimální řešení. Při popisu metod se zaměříme především na heuristické metody, neboť jsou časově efektivní a jejich řešení je pro zkoumané účely dostačující.

My zde ale uvedeme i metodu dynamického programování, která nalezne optimální řešení daného problem. Tuto metodu lze použít pro řešení „malých“ problémů. Pro řešení rozsáhlejších úloh je tato metoda z hlediska časové náročnosti nepoužitelná.

2.1.1 *Vybrané metody pro řešení úlohy obchodního cestujícího*

Existuje velké množství metod pro řešení úlohy obchodního cestujícího. V tomto paragrafu popíšeme takové metody, které jsou použitelné pro řešení reálných dopravních problémů.

2.1.1.1 *Dynamické programování*

Diskrétní dynamické programování je odvětví optimalizace. Jedná se o efektivní techniku pro výpočet optimální strategie určitých rozhodovacích procesů. Hlavní myšlenka metody je založena na Bellmanově principu maxima, aneb že každá podstrategie optimální strategie je opět optimální.

Metoda je obzvláště vhodná pro řešení „malých problémů“, které se dají rozložit na jednodušší podproblémy [8].

Metodu dynamického programování formuloval v roce 1953 Richard Bellman ve svém článku „*The theory of dynamic programming*“[9], kde mimo jiné uvedl dnes velmi známou Bellmanovu rovnici. Jedná se o vztah, kdy výpočet daného problému převedeme na postupné řešení „menších“ problémů.

Tato technika byla vybrána pro detailnější popis v rámci této práce, neboť jako jediná z popisovaných metod zajistí opravdu optimální řešení problem obchodního cestujícího. Není sice vhodná pro řešení „velkých” problémů, nicméně pro úlohy, které mají počet destinací méně než 20, je tato technika velice efektivní. Přesnější odhad destinací, pro který je technika použitelná bude stanoven později v kapitole 4.

Indexujme destinace jako $1,2,3,\dots,N$, kde $N \in \mathbb{N}$ je počet destinací. Bez újmy na obecnosti chápejme destinaci s indexem 1 jako startovní a zároveň cílové místo. Pokud by tento předpoklad nebyl splněn, destinace lze přeindexovat.

Dále mějme matici vzdáleností $C_{ij} \in \mathbb{R}_0^+$, kde C_{ij} je vzdálenost mezi i-tou a j-tou destinací, $i,j \in \{1,2,3,\dots,N\}, C_{ij} = 0$ pro $i = j$. Pro každou podmnožinu $S \subseteq \{1,2,3,\dots,N\}$ s vlastností $\{1,j\} \in S$ označme $C(S,j)$ jako délku nejkratší cesty, která prochází každou destinaci z S právě jednou, startuje v destinaci 1 a končí v destinaci j.

Rozdělme nyní problém nalezení nejkratší cesty dle definice $C(S,j)$ na dílčí podproblémy. Trasa musí být dle definice úlohy obchodního cestujícího cyklická. Startovní a zároveň cílová destinace je označena jako 1 a předposlední destinaci na trase označme jako $j \in S$. Pokud označíme destinaci, která v okruhu přímo předchází j-té destinaci jako $i \in S, i \neq j$, pak lze nahlédnout, že $C(S,j)$ se skládá z délky nejkratší cesty do i, což lze vyjádřit jako $C(S \setminus \{j\}, i)$, a ze vzdálenosti d_{ij}. Platí následující rovnost:

$$C(S,j) = \min_{i \in S, i \neq j, i \neq 1} \{C(S \setminus \{j\}, i) + d_{ij}\}. \qquad \text{(R 2.1)}$$

Pro $|S| = 2$ je řešení triviální, neboť pro $S = \{1,j\}$ platí $C(S,j) = d_{1j}$.

Mezivýsledky zpravidla zaznamenáváme do tabulek. Pro každou velikost množiny S vytvoříme novou tabulku. Řádky označujeme konkrétní množinou S a sloupce indexujeme dle $j \in S$. Při výpočtu pak snadno získáme $C(S,j)$ na základě mezivýsledku $C(S \setminus \{j\}, i)$ a d_{ij} (viz paragraf 2.1.1.1.2).

2.1.1.1.1 Popis algoritmu

Při popisu algoritmu se budeme držet značení popsaného výše. Algoritmus v pseudokódu lze vyjádřit následovně:

$for\ u \in \{2,3,\dots,N\}$:

$C(\{1,u\},u) = d_{1u}$

$for\ k = 3\ to\ N$

$for\ S \subseteq \{1,2,3,\dots,N\}; |S| = k; 1 \in S$:

$for\ j \in S; j \neq 1$:

$C(S,j) = \min_{i \in S, i \neq j, i \neq 1} \left(C(S \setminus \{j\}, i) + d_{ij}\right)$

$return\ \min_{i \in \{2,3,\dots,N\}} (C(\{1,2,3,\dots,N\}, i) + d_{i1})$.

Existuje nejvýše $2^n \cdot n$ podproblémů a každý z nich je řešitelný v lineárním čase. Celková časová složitost je proto $O(n^2 2^n)$.

2.1.1.1.2 Příklad

Uvažujme úlohu obchodního cestujícího pro 5 destinací, které budeme indexovat jako $1,2,3,4,5$. Dále mějme matici vzdáleností mezi jednotlivými destinacemi (Tab. 2.1), kde vzdálenost mezi i-tou a j-tou destinací je hodnota buňky v i-tém řádku a j-tém sloupci, $i, j \in \{1,2,3,4,5\}$.

Bez újmy na obecnosti předpokládejme, že matice je symetrická, neboť postup výpočtu v případě nesymetrické matice je shodný. Startovní a cílovou destinaci indexujeme jako 1. Pokud tento předpoklad není splněn, lze jednoduše přeindexovat destinace.

V následujícím příkladě se budeme držet značení definovaného výše.

$k\backslash k$	1	2	3	4	5
1	0	6	8	11	10
2		0	9	15	16
3			0	9	16
4				0	13
5					0

Tab. 2.1 Matice vzdáleností mezi destinacemi v ilustračním příkladě problému obchodního cestujícího

Cílem úlohy je určit optimální cyklickou trasu, jejíž startovní i cílovou destinací je 1, přičemž každou destinaci (mimo destinaci 1) navštíví obchodní cestující právě jednou. Optimální v tomto smyslu chápeme takovou trasu, kdy součet vzdáleností mezi jednotlivými vrcholy trasy je minimální možný.

Postupujme dle algoritmu metody dynamického programování.

Nejprve si vytvoříme tabulku, kde zaznamenáme hodnoty pro $C(S,j)$, kdy $|S| = 2$. Dle popisu algoritmu platí $C(S,k) = d_{1k}$.

$S\backslash k$	2	3	4	5
$\{1,k\}$	6	8	11	10

Tab. 2.2 Tabulka hodnot pro |S|=2 v ilustračním příkladě metody dynamického programování

Nyní vytvoříme novou tabulku, tentokrát pro hodnoty $C(S,j)$, kde $|S| = 3$. Pro S lze uvažovat tyto množiny: $\{1,2,k\}$, $\{1,3,k\}$, $\{1,4,k\}$ a $\{1,5,k\}$. Při výpočtu $C(S,j)$ pro tyto množiny postupujeme dle algoritmu a využíváme hodnot vypočtených v předchozí tabulce. Například $C(\{1,2,k\},k) = C(\{1,2\},2) + d_{2k} = 6 + d_{2k}$.

$S\backslash k$	2	3	4	5
$\{1,2,k\}$		15	21	22
$\{1,3,k\}$	17		23	24
$\{1,4,k\}$	26	20		24
$\{1,5,k\}$	26	26	23	

Tab. 2.3 Tabulka hodnot pro |S|=3 v ilustračním příkladě metody dynamického programování

Počet množin pro další tabulky se exponenciálně zvyšuje, proto zavedeme speciální značení, které zredukuje velikost tabulky. Pro množinu $\mathcal{A} \subseteq S$ budeme značit $P(\mathcal{A})$ jako množinu permutací všech prvků $x \in \mathcal{A}$. Pro množinu $\mathcal{A} = \{1,2,3\}$ tedy chápeme $P(\mathcal{A}) = \{\{1,2,3\},\{1,3,2\},\{2,1,3\},\{2,3,1\},\{3,1,2\},\{3,2,1\}\}$. Toto značení budeme používat v rámci množin S. Množiny $\{\{1,2,3,k\},\{1,3,2,k\}\}$ tedy vyjádříme jako $\{1,P(\{2,3\}),k\}$. Dále budeme značit $min(x,y)$ jako minimum z hodnot x a y.

Další tabulku sestrojíme pro hodnoty $C(S,j)$, kde $|S| = 4$.

Pro S lze uvažovat tyto množiny: $\{1,P(\{2,3\}),k\}$, $\{1,P(\{2,4\}),k\}$, $\{1,P(\{2,5\}),k\}$, $\{1,P(\{3,4\}),k\}$, $\{1,P(\{3,5\}),k\}$, $\{1,P(\{4,5\}),k\}$.

Při výpočtu $C(S,j)$ pro tyto množiny opět postupujeme dle algoritmu a využíváme hodnot vypočtených v předchozí tabulce.

Například $C(\{1,P(\{2,3\}),k\},k) = \min\big(C(\{1,2,3,k\},k); C(\{1,3,2,k\},k)\big)$, kde $C(\{1,2,3,k\},k) = C(\{1,2,3\},3) + d_{3k} = 15 + d_{3k}$ a $C(\{1,3,2,k\},k) = C(\{1,3,2\},2) + d_{2k} = 17 + d_{2k}$, tudíž $C(\{1,P(\{2,3\}),k\},k) = min(2 + d_{3k}; 4 + d_{2k})$.

$S\backslash k$	2	3	4	5
$\{1,P(\{2,3\}),k\}$			24	31
$\{1,P(\{2,4\}),k\}$		30		34
$\{1,P(\{2,5\}),k\}$		35	35	
$\{1,P(\{3,4\}),k\}$	29			36
$\{1,P(\{3,5\}),k\}$	35		35	
$\{1,P(\{4,5\}),k\}$	38	32		

Tab. 2.4 Tabulka hodnot pro |S|=4 v ilustračním příkladě metody dynamického programování

Podobně postupujeme i při výpočtu $C(S,j)$ pro $|S| = 5$. Z důvodu rozsahu práce uvedeme pouze finální tabulku.

$S\backslash k$	2	3	4	5
$\{1, P(\{2,3,4\}), k\}$				37
$\{1, P(\{2,3,5\}), k\}$			44	
$\{1, P(\{2,4,5\}), k\}$		44		
$\{1, P(\{3,4,5\}), k\}$	41			

Tab. 2.5 Tabulka hodnot pro |S|=5 v ilustračním příkladě metody dynamického programování

Posledním krokem algoritmu je určení předposlední destinace v rámci trasy. Tu určíme tak, že vypočteme minimum z hodnot $(C(\{1, P(\{2,3,4\}), 5\}, 5) + d_{51})$, $(C(\{1, P(\{2,3,5\}), 4\}, 4) + d_{41})$, $(C(\{1, P(\{2,4,5\}), 3\}, 3) + d_{31})$ a $(C(\{1, P(\{3,4,5\}), 2\}, 2) + d_{21})$. Získáme hodnotu 47, což je cena (suma vzdáleností) optimální cyklické trasy, která startuje a končí v destinaci 1 a každý vrchol navštíví právě jednou.

Přesné pořadí vrcholů v optimální trase lze snadno zpětně zkonstruovat z tabulek hodnot. Pro konkrétní hodnoty matice vzdáleností mezi destinacemi jsme nalezli více variant optimální trasy: $\{1,2,3,4,5,1\}$ a $\{1,5,4,3,2,1\}$. Díky symetričnosti matice vzdáleností byly vypočteny trasy navzájem opačné.

2.1.1.2 *Metoda Clarke-Wrighta*

Jedná se o jednu z nejznámějších heuristických metod pro řešení úlohy obchodního cestujícího, kterou v roce 1964 popsali Clarke a Wright [10]. Metodu lze použít i pro zobecněné úlohy a další varianty dopravních problémů [11].

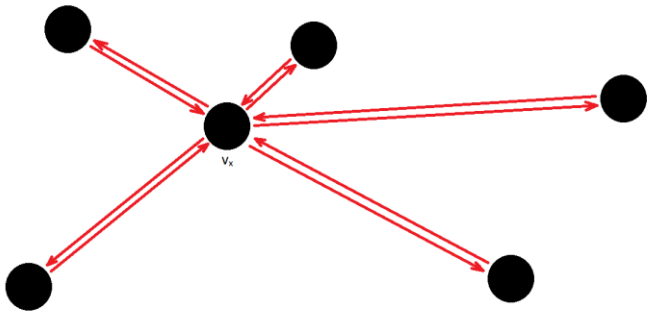
Metoda byla vybrána pro svou implementační a časovou efektivitu. V rámci porovnání známých metod pro řešení okružního dopravního problému, které provedl RNDr. Petr Kučera ve své práci Metodologie řešení okružního dopravního problému, lze předpokládat, že tato metoda se zdá být vhodná pro vyhledání efektivních řešení „velkých" problemů [12].

Do „výhodnostního koeficientu" (definován níže) lze také zakomponovat některá další omezení v rámci objednávky, příp. vozidla, a jednoduše tak metodu přizpůsobit konkrétnímu problému.

Uvažujme úlohu obchodního cestujícího jako graf $G = (V; E)$, kde $V = \{v_1, v_2, \ldots, v_N\}$ jsou destinace a E hrany mezi nimi, $N \in \mathbb{N}$ je počet destinací. Ohodnocení hran $H: E \to \mathbb{R}_0^+$ budeme značit jako $h\left((v_i; v_j)\right)$ pro hranu $(v_i; v_j) \in E$. Při popisu budeme značit trasu z v_i do v_j jako $v_i \to v_j$. Zavedeme termín „cena okruhu“, což lze chápat jako součet vzdáleností všech tras daného okruhu.

V prvním kroku algoritmu je náhodně zvolena jedna destinace $v_x \in V$, kterou budeme dále nazývat výchozí. Touto destinací můžeme chápat sklad nebo továrnu, ze které rozvážíme zboží do cílových destinací. Tato destinace je tedy startovním i cílovým místem okruhu. Zavedeme také pomocné struktury $\mathcal{A}, \mathcal{B} \subseteq V$, které budou charakterizovat vrcholy, do nichž vede trasa z výchozí destinace (v případě $\mathcal{A}$), resp. z nichž vede trasa do výchozí destinace (v případě $\mathcal{B}$). Na počátku volíme $\mathcal{A} = \mathcal{B} = \emptyset$.

Nyní zkonstruujeme elementární řešení problému, tzn. pro každé $v_i \in V, v_i \neq v_x$ přidáme do okruhu T trasu $v_x \to v_i$ a $v_i \to v_x$ (viz Obr. 2.1) a přidáme v_i do množiny vstupních krajních vrcholů $\mathcal{A}$ i do množiny výstupních krajních vrcholů $\mathcal{B}$. Nyní je tedy $\mathcal{A} = \mathcal{B} = V \setminus \{v_x\}$.



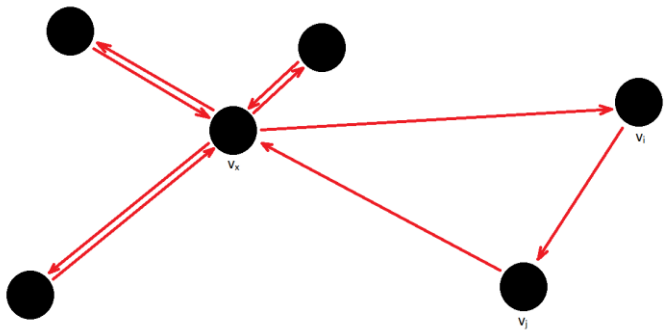
Obr. 2.1 Elementární řešení metody Clarke-Wrighta v ilustračním příkladě úlohy obchodního cestujícího

V dalším kroku metody určíme dvojici destinací (v_i, v_j), kde $v_i \neq v_x$, $v_j \neq v_x$ a $v_i \neq v_j$. Samotný výběr je založen na myšlence prioritního řazení takových dvojic destinací, kde vzdálenost mezi těmito destinacemi je výrazně nižší než vzdálenost každé z nich do destinace výchozí. Tento poměr vzdáleností charakterizujeme pomocí tzv. „výhodnostního koeficientu“ z_{ij} pro i-tou a j-tou destinaci:

$$z_{ij} = h\big((v_x; v_i)\big) + h\Big(\big(v_j; v_x\big)\Big) - h\Big(\big(v_i; v_j\big)\Big),$$
$$i, j \in \{1,2, \dots, N\}; i, j \neq x; i \neq j. \qquad \text{(R 2.2)}$$

Spočítáme výhodnostní koeficienty pro všechny dvojice destinací (v_i, v_j), $i, j \in \{1, \dots, N\}; i, j \neq x; i \neq j$. Seřadíme dvojice destinací dle výhodnostních koeficientů sestupně. Nyní vybereme dvojici (v_i, v_j) s nejvyšším kladným výhodnostním koeficientem, kde $v_j \in \mathcal{A}, v_i \in \mathcal{B}$. Zároveň musí být splněna podmínka, že odebráním tras $v_i \to v_x$, $v_x \to v_j$ z okruhu T a přidáním trasy $v_i \to v_j$ do okruhu T nevznikne nový okruh, který by neobsahoval výchozí vrchol $v_x \in V$. Pokud taková dvojice neexistuje, pak skončíme a výsledek prohlásíme za výsledné řešení.

Odebereme trasy $v_i \to v_x$, $v_x \to v_j$ a přidáme trasu $v_i \to v_j$ (viz Obr. 2.2). Odebereme v_i z množiny výstupních krajních vrcholů $\mathcal{B}$ a v_j z množiny vstupních krajních vrcholů $\mathcal{A}$.



Obr. 2.2 Přepojení tras pro dvojici destinací v_i a v_j v ilustračním příkladě úlohy obchodního cestujícího

Nyní opět vybíráme další dvojici (v_i, v_j) s nejvyšším výhodnostním koeficientem, kde $v_j \in \mathcal{A}$ a $v_i \in \mathcal{B}$ tak, aby odebráním tras $v_i \to v_x$, $v_x \to v_j$ z okruhu T a přidáním trasy $v_i \to v_j$ do okruhu T nevznikl nový okruh, který by neobsahoval výchozí vrchol $v_x \in V$. Pokud taková dvojice existuje, opakujeme postup s přepojováním tras zmíněný v odstavci výše, jinak skončíme a okruh prohlásíme za výsledné řešení.

Výsledné řešení zkonstruované metodou Clarke-Wright může být složeno z více okruhů, z nichž každý začíná i končí ve výchozím vrcholu $v_x \in V$. Je to způsobeno tím, že výhodnostní

koeficient pro určitou dvojici vrcholů může být i záporný. Přepojení tras dle této dvojice vrcholů je nežádoucí, neboť by zvýšilo cenu okruhu.

2.1.1.2.1 Popis algoritmu

Při popisu algoritmu se budeme držet značení definovaného výše.

I. Náhodně zvolíme výchozí uzel $v_x \in V$.Vytvoříme okruh $T = \{v_x \to v_i, v_i \to v_x\}$, $v_i \in V$, $i \in \{1,2,\dots,N\}$, $i \neq x$. $\mathcal{A} = \mathcal{B} = V \setminus \{v_x\}$.

II. Spočítáme výhodnostní koeficienty z_{ij} pro všechny dvojice destinací (v_i, v_j), kde $i, j \in \{1,2,\dots,N\}; i, j \neq x; i \neq j$:

$$z_{ij} = h\big((v_x; v_i)\big) + h\Big(\big(v_j; v_x\big)\Big) - h\Big(\big(v_i; v_j\big)\Big). \qquad \text{(R 2.3)}$$

III. Vybereme dvojici (v_i, v_j) s nejvyšším kladným výhodnostním koeficientem, kde $v_j \in \mathcal{A}$ a $v_i \in \mathcal{B}$ a zároveň odebráním tras $v_i \to v_x$, $v_x \to v_j$ z okruhu T a přidáním trasy $v_i \to v_j$ do okruhu T nevznikl nový okruh, který by neobsahoval výchozí vrchol $v_x \in V$. Pokud taková dvojice neexistuje, pak skončíme, a výsledek prohlásíme za výsledné řešení.

IV. $T := T \setminus \{v_i \to v_x, v_x \to v_j\} \cup \{v_i \to v_j\}$. $\mathcal{A} := \mathcal{A} \setminus \{v_j\}$, $\mathcal{B} := \mathcal{B} \setminus \{v_i\}$. Opakujeme krok III.

2.1.1.2.2 Příklad

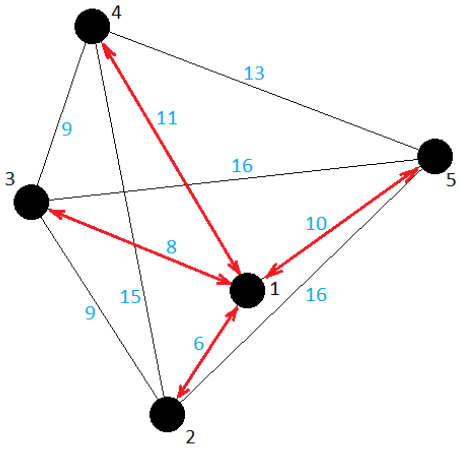
Uvažujme stejný příklad jako v paragrafu 2.1.1.1.2. Mějme úlohu obchodního cestujícího pro 5 destinací, které budeme indexovat jako $1,2,3,4,5$. Dále mějme matici vzdáleností mezi jednotlivými destinacemi (Tab. 2.1), kde vzdálenost mezi i-tou a j-tou destinací je hodnota buňky v i-tém řádku a j-tém sloupci, $i, j \in \{1,2,3,4,5\}$.

Bez újmy na obecnosti předpokládejme, že matice je symetrická, neboť postup výpočtu v případě nesymetrické matice je shodný. Dále startovní a cílovou destinaci indexujeme jako 1. Pokud tento předpoklad není splněn, lze jednoduše přeindexovat destinace.

V následujícím příkladě se budeme držet značení definovaného výše.

Postupujme dle metody Clarke-Wrighta.

Startovní a zároveň cílovou destinací zvolíme destinaci 1 a vytvoříme elementární řešení. Přidáme trasy $v_1 \rightarrow v_k$ a $v_k \rightarrow v_1$ pro každé $k \in \{2,3,4,5\}$ do okruhu T a přidáme v_k do množiny vstupních vrcholů $\mathcal{A}$ i do množiny výstupních vrcholů $\mathcal{B}$. Okruh T tedy nyní obsahuje trasy: $v_1 \rightarrow v_2$, $v_2 \rightarrow v_1$, $v_1 \rightarrow v_3$, $v_3 \rightarrow v_1$, $v_1 \rightarrow v_4$, $v_4 \rightarrow v_1$, $v_1 \rightarrow v_5$, $v_5 \rightarrow v_1$, $\mathcal{A} = \mathcal{B} = \{2,3,4,5\}$. Tento okruh má cenu 70 $(6 + 6 + 8 + 8 + 11 + 11 + 10 + 10)$.



Obr. 2.3 Elementární řešení metody Clarke-Wrighta v ilustračním příkladě úlohy obchodního cestujícího

Spočítáme výhodnostní koeficienty pro všechny dvojice destinací (v_i, v_j), $i, j \in \{2,3,4,5\}; i \neq j$ dle následujícího vztahu:

$$z_{ij} = h\big((v_x; v_i)\big) + h\Big(\big(v_j; v_x\big)\Big) - h\Big(\big(v_i; v_j\big)\Big) \qquad \text{(R 2.4)}$$

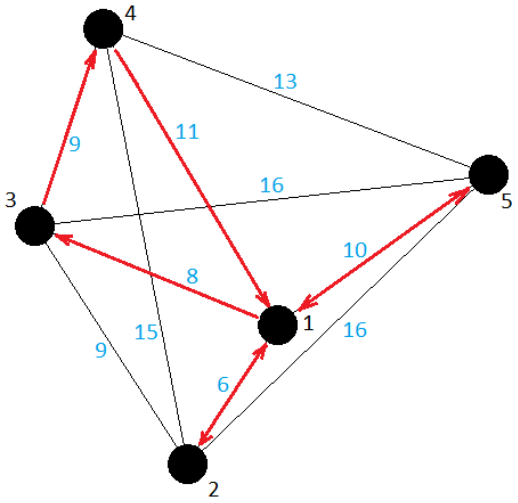
Tyto výhodnostní koeficienty zaznamenáme do tabulky (Tab. 2.6). Vzhledem k tomu, že matice vzdáleností (**Chyba! Nenalezen zdroj odkazů.**) je symetrická, nezáleží na pořadí ve dvojici destinací pro výpočet výhodnostního koeficientu.

	z_{ij}
$(v_2; v_3)$	5
$(v_3; v_2)$	5
$(v_2; v_4)$	2
$(v_4; v_2)$	2
$(v_2; v_5)$	0
$(v_5; v_2)$	0
$(v_3; v_4)$	10
$(v_4; v_3)$	10
$(v_3; v_5)$	2
$(v_5; v_3)$	2
$(v_4; v_5)$	8
$(v_5; v_4)$	8

Tab. 2.6 Tabulka výhodnostních koeficientů pro ilustrační příklad úlohy obchodního cestujícího

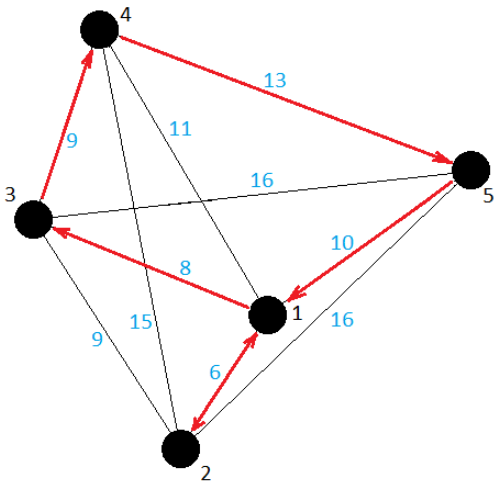
Nyní vybereme dvojici (v_i, v_j) s nejvyšším kladným výhodnostním koeficientem, kde $v_j \in \mathcal{A} = \{2,3,4,5\}, v_i \in \mathcal{B} = \{2,3,4,5\}$. Zároveň musí být splněna podmínka, že odebráním tras $v_i \to v_x$, $v_x \to v_j$ a přidáním trasy $v_i \to v_j$ nevznikne nový okruh, který by neobsahoval výchozí vrchol 1.

Nejvyššího výhodnostního koeficientu 10 dosáhly dvě dvojice destinací $(v_3; v_4)$ a $(v_4; v_3)$. Nehrozí, že přepojením tras dle jedné z těchto dvojic vznikne nový okruh bez výchozí destinace 1. Z uvažovaných dvojic náhodně vybereme dvojici $(v_3; v_4)$. Odebereme z okruhu T trasy $v_3 \to v_1$, $v_1 \to v_4$ a přidáme trasu $v_3 \to v_4$. Odebereme v_3 z množiny výstupních vrcholů $\mathcal{B}$ a v_4 z množiny vstupních vrcholů $\mathcal{A}$. Okruh tedy nyní obsahuje trasy: $v_1 \to v_2$, $v_2 \to v_1$, $v_1 \to v_3$, $v_4 \to v_1$, $v_1 \to v_5$, $v_5 \to v_1$, $v_3 \to v_4$. Tento okruh má cenu 60 $(6 + 6 + 8 + 9 + 11 + 10 + 10)$.



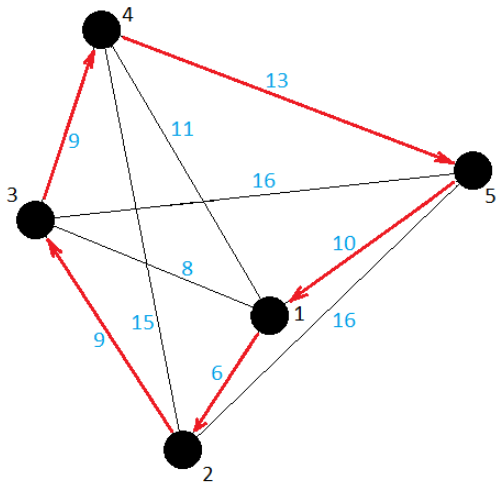
Obr. 2.4 Řešení metody Clarke-Wright po první iteraci v ilustračním příkladě úlohy obchodního cestujícího

Opakujeme postup vybírání dvojice destinací s nejvyšším výhodnostním koeficientem, tentokrát pouze mezi dvojicemi $(v_i; v_j)$, kde $v_i \in \mathcal{B} = \{v_2, v_4, v_5\}$ a $v_j \in \mathcal{A} = \{v_2, v_3, v_5\}$. Nejvyššího výhodnostního koeficientu 8 dosáhla dvojice destinací $(v_4; v_5)$. Opět nehrozí, že přepojením tras dle této dvojice vznikne nový okruh bez výchozí destinace 1. Odebereme z okruhu T trasy $v_4 \to v_1$, $v_1 \to v_5$ a přidáme trasu $v_4 \to v_5$. Odebereme v_4 z množiny výstupních vrcholů $\mathcal{B}$ a v_5 z množiny vstupních vrcholů $\mathcal{A}$. Okruh tedy nyní obsahuje trasy: $v_1 \to v_2$, $v_2 \to v_1$, $v_1 \to v_3$, $v_5 \to v_1$, $v_3 \to v_4$, $v_4 \to v_5$. Tento okruh má cenu 52 $(6 + 6 + 8 + 9 + 13 + 10)$.



Obr. 2.5 Řešení metody Clarke-Wrighta po druhé iteraci v ilustračním příkladě úlohy obchodního cestujícího

Opět opakujeme postup vybírání dvojice destinací s nejvyšším výhodnostním koeficientem, tentokrát pouze mezi dvojicemi $(v_i; v_j)$, kde $v_i \in \mathcal{B} = \{v_2, v_5\}$ a $v_j \in \mathcal{A} = \{v_2, v_3\}$. Nejvyšší výhodnostní koeficient 5 má dvojice $(v_2; v_3)$. Opět nehrozí, že přepojením tras dle této dvojice vznikne nový okruh bez výchozí destinace 1. Odebereme z okruhu T trasy $v_2 \rightarrow v_1$, $v_1 \rightarrow v_3$ a přidáme trasu $v_2 \rightarrow v_3$. Odebereme v_2 z množiny výstupních vrcholů $\mathcal{B}$ a v_3 z množiny vstupních vrcholů $\mathcal{A}$. Okruh tedy nyní obsahuje trasy: $v_1 \rightarrow v_2$, $v_5 \rightarrow v_1$, $v_3 \rightarrow v_4$, $v_4 \rightarrow v_5$, $v_2 \rightarrow v_3$. Tento okruh má cenu 47 $(6 + 9 + 9 + 13 + 10)$.



Obr. 2.6 Řešení metody Clarke-Wrighta po třetí iteraci v ilustračním příkladě úlohy obchodního cestujícího

Opět opakujeme postup vybírání dvojice, tentokrát pouze mezi dvojicemi $(v_i; v_j)$, kde $v_i \in \mathcal{B} = \{v_5\}$ a $v_j \in \mathcal{A} = \{v_2\}$. Neexistuje taková dvojice destinací s kladným výhodnostním koeficientem. Pokud bychom vybrali dvojici destinací $(v_5; v_2)$, která má nulový výhodnostní koeficient, pak by vznikl okruh, který neobsahuje výchozí vrchol 1.

Poslední získané řešení prohlásíme za finální. Cena okruhu je tedy 47 $(6 + 9 + 9 + 13 + 10)$. Díky výpočtu v paragrafu 2.1.1.1.2 také víme, že se jedná o optimální řešení úlohy obchodního cestujícího.

2.1.2 *Ostatní metody*

V rámci tohoto paragrafu popíšeme další známé metody pro řešení úlohy obchodního cestujícího. Zaměříme se především na heuristické metody, neboť jsou časově efektivní a jejich řešení je pro zkoumané účely dostačující.

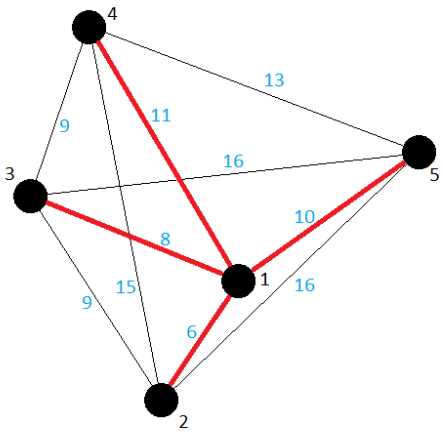
2.1.2.1 *Metody minimální kostry*

V tomto paragrafu uvedeme několik metod, které využívají minimální kostru ohodnoceného grafu. Problém minimální kostry ohodnoceného grafu lze řešit např. Kruskalovým algoritmem [13]. Efektivnější algoritmus popsal v roce 1926 Otakar Borůvka[14], nebo později roku 1930 Vojtěch Jarník [15]. Dosud nejrychlejší známý deterministický algoritmus pro hledání minimální kostry grafu sestrojil Bernard Chazelle v roce 2000 modifikací Borůvkova algoritmu [16]. Asymptotická složitost tohoto algoritmu je $O\big(|E|\alpha(|V|)\big)$ kde α je Ackermannova funkce, V množina vrcholů a E množina hran.

Uvažujme úlohu obchodního cestujícího jako graf $G = (V; E)$, kde $V = \{v_1, v_2, \dots, v_N\}$ jsou destinace a E hrany mezi nimi, $N \in \mathbb{N}$ je počet destinací. Ohodnocení hran $H: E \to \mathbb{R}_0^+$ budeme značit jako $h\left(\left(v_i; v_j\right)\right)$ pro hranu $\left(v_i; v_j\right) \in E$. Následující metody jsou založeny na předpokladu metriky, tzn. musí platit trojúhelníková nerovnost:

$$h(v_i, v_k) \leq h\big(v_i, v_j\big) + h\big(v_j, v_k\big), \quad i, j, k \in \{1, 2, \dots, N\}. \tag{R 2.5}$$

Metody nejdříve zkonstruují minimální kostru grafu pomocí některé z výše zmíněných metod v [13][14][15][16]. Pro ilustraci uvádíme minimální kostru grafu (Obr. 2.7) pro příklad úlohy obchodního cestujícího řešený v paragrafech 2.1.1.1.2 a 2.1.1.2.2.



Obr. 2.7 Minimální kostra grafu v ilustračním příkladě úlohy obchodního cestujícího

Z definice minimální kostry plyne, že ohodnocení minimální kostry je nižší než ohodnocení jakékoli cyklické trasy v grafu, která obsahuje všechny vrcholy, neboť cyklické řešení obsahuje alespoň o jednu hranu více než minimální kostra. Řešení úlohy obchodního cestujícího má tedy větší ohodnocení než minimální kostra daného grafu, neboť řešením úlohy obchodního cestujícího je cyklus.

Následující paragrafy konkretizují metody dle postupu získávání řešení úlohy obchodního cestujícího z minimální kostry.

2.1.2.1.1 2-aproximační algoritmus

Tato varianta využívá algoritmu průchodu grafu do hloubky („Depth-first search“), který lze nalézt např. v [17].

Zvolíme náhodně vrchol $v_x \in V$ a projdeme graf z vrcholu $v_x \in V$ dle algoritmu průchodu grafu do hloubky. Obecně je výhodné zvolit takový vrchol $v_x \in V$, který má nejvyšší stupeň. Při průchodu grafu do hloubky si zaznamenáváme všechny průchody vrcholů. Dle algoritmu je zřejmé, že po skončení výpočtu můžeme mít některé vrcholy zaznamenány vícekrát.

Pokud v našem případě provedeme průchod grafu do hloubky např. z vrcholu 1 dle (Obr. 2.7), pak můžeme získat následující posloupnost vrcholů $\{1, 2, 1, 3, 1, 4, 1, 5\}$.

V dalším kroku algoritmu projdeme zaznamenané vrcholy a odstraníme duplicity, tj. ponecháme pouze první výskyt každého vrcholu. V konkrétním případě dle (Obr. 2.7) dostáváme posloupnost $\{1,2,3,4,5\}$. Tímto krokem dojde k vytvoření posloupnosti vrcholů pro výsledné cyklické řešení a díky platnosti trojúhelníkové nerovnosti je ohodnocení nově vzniklého řešení maximálně dvojnásobné oproti původní minimální kostře [18].

Výsledné řešení úlohy obchodního cestujícího pro vytvořenou posloupnost $\{v_{m_1}, v_{m_2}, \dots, v_{m_N}\}$ získáme tak, že do okruhu přidáme trasy

$$v_{m_i} \rightarrow v_{m_{(i+1)\ mod\ N}}, \quad i \in \{1,2, \dots, N\}. \tag{R 2.6}$$

Metodu lze vylepšit tak, že postup opakujeme pro všechny volby vrcholu $v_x \in V$ a volíme takové $v_x \in V$, pro které je suma vzdáleností mezi vrcholy výsledného okruhu T nejmenší.

Detailní popis výpočtu neuvádíme, nicméně 2-aproximační algoritmus aplikovaný na úlohu v paragrafu 2.1.1.1.2 vydá posloupnost vrcholů v okruhu $\{1,2,3,4,5\}$ a cena okruhu je 47.

Díky výpočtu příkladu v paragrafu 2.1.1.1.2 také víme, že se jedná o optimální řešení úlohy obchodního cestujícího.

2.1.2.1.2 Christofidesův algoritmus

Tato varianta využívá konstrukce eulerovského tahu, jejíž popis lze nalézt v [17]. Minimální kostru grafu $G = (V; E)$ budeme značit $\acute{G} = (V; \acute{E})$, kde $\acute{E} \subseteq E$. Algoritmus popíšeme ve 4 krocích:

I. Vytvoříme úplný graf H na množině všech vrcholů, které v kostře $\acute{G}$ mají lichý stupeň.
II. V grafu H najdeme nejlevnější perfektní párování $\mathcal{P}$ dle ohodnocení H.
III. Hrany $\mathcal{P}$ přídáme k hranám $\acute{E}$ minimální kostry. Graf $(V, \mathcal{P} \cup \acute{E})$ je eulerovský graf. V grafu $(V, \mathcal{P} \cup \acute{E})$ sestrojíme uzavřený eulerovský tah.
IV. Trasu T získáme z eulerovského tahu tak, že vrcholy navštívíme v pořadí, ve kterém jsme do nich poprvé vstoupili při tvorbě eulerovského tahu.

Platí, že délka trasy T je maximálně $\frac{3}{2}$ krát větší než délka optimální trasy. Efektivita je zde vykoupena výrazně složitější implementací a na reálných datech se dále ukazuje, že výsledek není v průměru o mnoho lepší, než při použití 2-aproximačního algoritmu [18].

Detailní popis výpočtu příkladu z paragrafu 2.1.1.1.2 zde neuvádíme. Pro porovnání s ostatními algoritmy uvádíme jen výsledek. Christofidesův algoritmus vydá posloupnost vrcholů v okruhu $\{1,2,3,1,4,5\}$ a cena tohoto okruhu je 57.

2.1.2.2 *Metoda zatřiďování*

Metoda zatřiďování („Nearest Merger Method“) je další z řady metod, které využívají obecné znalosti teorie grafů. Lze ji nalézt např. v publikaci p. Kučery [19].

Uvažujme problém obchodního cestujícího jako graf $G = (V; E)$, kde $V = \{v_1, v_2, \dots, v_N\}$ jsou destinace, $N \in \mathbb{N}$ je počet destinací a E množina hran mezi nimi. Ohodnocení hran $H: E \to \mathbb{R}_0^+$ budeme značit jako $h\left((v_i; v_j)\right)$ pro hranu $(v_i; v_j) \in E$. K jednotlivým vrcholům přidáme informaci o tom, do jakého patří okruhu. Okruh chápejme jako cyklickou trasu procházející přes určité vrcholy.

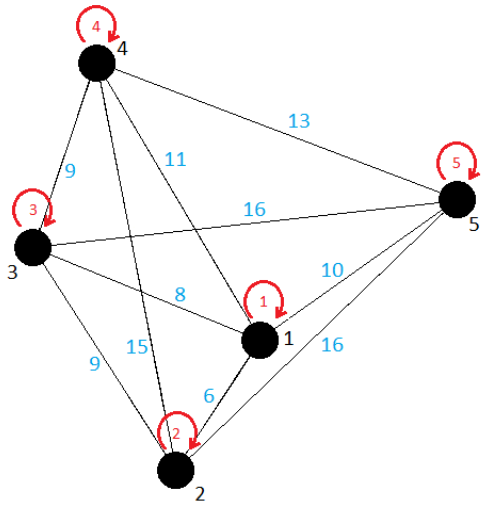
Metoda vychází z elementárního řešení, kdy pro každou destinaci je vytvořen okruh, který obsahuje pouze tuto destinaci. V každé iteraci algoritmu jsou spojeny dva okruhy do jednoho, tzn. je přidána trasa mezi vrcholem jednoho okruhu a vrcholem druhého okruhu. Algoritmus končí ve stavu, kdy všechny vrcholy grafu $G = (V; E)$ náleží jedinému okruhu.

Při popisu budeme značit trasu z v_i do v_j jako $v_i \to v_j$. Pokud vrchol v_i je obsažen v v okruhu x, pak to označíme jako $v_i \in circle(x)$. Dále zavedeme porovnání vzdálenosti mezi okruhy. Budeme ho značit jako $(circle(x); circle(y))$ a definujeme jako minimální ohodnocení hrany pro vrcholy $v_i \in V$ a $v_j \in V$, kde $v_i \in circle(x)$ a $v_j \in circle(y)$. Formálně tedy

$$\begin{gathered} \big(circle(x); circle(y)\big) \coloneqq h\left(\left(v_i; v_j\right)\right), \\ h\left(\left(v_i; v_j\right)\right) \leq h\big((v_m; v_n)\big), \\ v_i, v_m \in circle(x), \end{gathered} \tag{R 2.7}$$

$$v_j, v_n \in circle(y).$$

Na začátku vytvoříme elementární řešení problému, tj. pro každý vrchol $v_i \in V$ přidáme trasu $v_i \to v_i$ do okruhu T a zaznamenáme, že tato trasa patří do i-tého okruhu ($(v_i \to v_i) \in circle(i)$). Tento krok vytvoří pro každý vrchol samostatný okruh. V našem příkladě z paragrafu 2.1.1.1.2 elementární řešení vypadá následovně:



Obr. 2.8 Elementární řešení metody zatřiďování v ilustračním příkladě úlohy obchodního cestujícího, červeným písmem značíme okruhy

Nyní budeme okruhy propojovat na základě vzdálenosti. Seřadíme okruhy dle indexů ($i \in \{1,2,\dots,N\}$) a postupně procházíme okruhy vzestupně. Vezmeme v pořadí první okruh e a najdeme okruh f takový, že platí vztah

$$\big(circle(e); circle(f)\big) \leq \big(circle(e); circle(g)\big), \qquad \text{(R 2.8)}$$

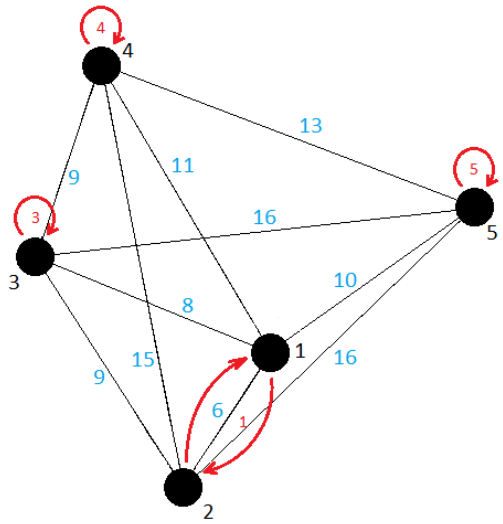
kde g indexuje všechny okruhy.

Najdeme dvojici tras $v_i \to v_j$ a $v_k \to v_l$ takovou, že $v_i, v_j \in circle(e)$ a $v_k, v_l \in circle(f)$ a zároveň pro všechny $v_m \to v_n$, kde $v_m, v_n \in circle(e)$ a $v_o \to v_p$, kde $v_o, v_p \in circle(f)$ platí

$$\begin{aligned}\Big(h\big((v_i; v_l)\big) + h\Big(\big((v_k; v_j)\big)\Big) - h\Big(\big((v_i; v_j)\big)\Big) - h\big((v_k; v_l)\big)\\ \leq h\Big(\big((v_m; v_p)\big)\Big) + h\big((v_o; v_n)\big) - h\big((v_m; v_n)\big) - h\Big(\big((v_o; v_p)\big)\Big)\Big).\end{aligned} \qquad \text{(R 2.9)}$$

Tento vztah lze chápat tak, že hledáme dvojici tras takovou, že při jejich propojení (a tedy spojení obou okruhů do jednoho) bude přírůstek ohodnocení vznikajícího okruhu nejmenší možný.

Nyní odebereme trasu $v_i \rightarrow v_j$ a $v_k \rightarrow v_l$ z okruhu T, přidáme trasy $v_i \rightarrow v_l$ a $v_k \rightarrow v_j$ do okruhu T a přeznačíme všechny trasy v okruhu f tak, aby nyní patřily do okruhu e, neboť došlo k propojení dvou okruhů (viz Obr. 2.9, zde spojení okruhu 1 a 2).



Obr. 2.9 Metoda zatřiďování po první iteraci v ilustračním příkladě úlohy obchodního cestujícího

Takto jsme spojili dva okruhy v jeden. Opakujeme celý postup pro další okruh v seznamu, který jsme na počátku třídili. Pokud již neexistuje žádný další okruh v seznamu, pokračujeme znovu od začátku seznamu vzestupně (tj. od okruhu s nejmenším indexem).

Algoritmus končí, když všechny vrcholy grafu $G = (V; E)$ náleží jedinému okruhu a tento okruh prohlásíme za finální řešení.

Metodu lze vylepšit především ve výběru okruhů ke spojování a v hledání tras k přepojování. Složitost metody také velkou měrou závisí na použitých strukturách při implementaci, reprezentaci grafu a dílčích okruhů.

2.1.2.2.1 Popis algoritmu

Při popisu budeme využívat značení definované výše.

I. Vytvoříme elementární řešení: $T = \{v_i \rightarrow v_i\}, v_i \in circle(i)$, $i \in \{1,2,..,N\}$ $O = \{1,2,\dots,N\}$, (viz Obr. 2.8).

II. Seřadíme okruhy O vzestupně, e nastavíme na první hodnotu seznamu O.

III. Vybereme okruh $circle(e)$ ze seřazeného seznamu a dále $circle(f)$ tak, aby $\forall(circle(g))\colon \big(circle(e); circle(f)\big) \leq \big(circle(e); circle(g)\big)$.

IV. Najdeme dvojici tras $v_i \rightarrow v_j$, $v_k \rightarrow v_l$, tak, že $v_i, v_j \in circle(e)$ a $v_k, v_l \in circle(f)$: $\Big(h\big((v_i; v_l)\big) + h\Big(\big(v_k; v_j\big)\Big) - h\Big(\big(v_i; v_j\big)\Big) - h\big((v_k; v_l)\big) \leq h\Big(\big(v_m; v_p\big)\Big) + h\big((v_o; v_n)\big) - h\big((v_m; v_n)\big) - h\Big(\big(v_o; v_p\big)\Big)\Big)$. Potom $T \coloneqq T \setminus \{v_i \rightarrow v_j, v_k \rightarrow v_l\} \cup \{v_i \rightarrow v_l, v_k \rightarrow v_j\}$, $\forall(v_m \in circle(f))\colon \big(v_m \in circle(e)\big)$, $O \coloneqq O \setminus \{f\}$.

V. Pokud $|O| > 1$, e nastavíme na bezprostředně následující vyšší hodnotu v seznamu O, pokud tato neexistuje, pak na nejnižší hodnotu seznamu O. Přejdeme na bod III. Pokud $|O| = 1$, skončíme a prohlásíme řešení za výsledné.

Detailní popis výpočtu příkladu z paragrafu 2.1.1.1.2 zde neuvádíme, pouze výsledek k porovnání s ostatními metodami. Metoda zatřiďování vydá posloupnost vrcholů v okruhu $\{1,2,1,3,4,1,5\}$ a cena tohoto okruhu je 60.

2.1.2.3 *Vkládací metody*

Bylo popsáno mnoho heuristik, které naleznou elementární okruh (většinou s jedinou a sice výchozí destinací) a poté do tohoto okruhu iterativně vkládají dosud nenavštívené destinace. S každou vloženou destinací tak rozšiřují okruh o další destinace, dokud okruh neobsahuje všechny destinace.

Tyto metody jsou obecně nazývány vkládacími a liší se způsobem vyhledání a přidání další destinace do okruhu. My se omezíme pouze na některé z těchto metod a to metodu nejbližšího přídavku, metodu nejbližšího vložení a metodu nejlevnějšího vložení.

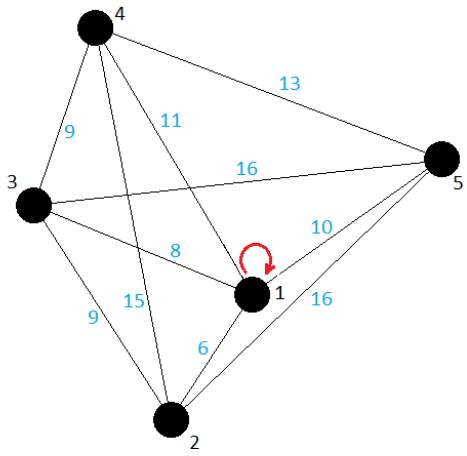
Uvažujme problém jako graf $G = (V; E)$, kde $V = \{v_1, v_2, \dots, v_N\}$ jsou destinace, $N \in \mathbb{N}$ je počet destinací a E hrany mezi nimi. Ohodnocení hran $H\colon E \rightarrow \mathbb{R}_0^+$ budeme značit jako

$h\left((v_i; v_j)\right)$ pro hranu $(v_i; v_j) \in E$. Při popisu budeme značit trasu z v_i do v_j jako $v_i \to v_j$ a zavedeme $\mathcal{A}$ jako pomocný seznam použitých vrcholů v okruhu. Na počátku je $\mathcal{A} = \emptyset$.

Pro okruh $\acute{T}$, jež obsahuje vrcholy $\acute{V} \subseteq V$, zavedeme speciální značení vzdálenosti vrcholu v_m od okruhu $\acute{T}$:

$$\begin{gathered} (v_m; \acute{T}) \coloneqq h((v_m; v_i)), \\ \forall (v_j \in \acute{V}): \left(h((v_m; v_i)) \leq h\left((v_m; v_j)\right)\right), \\ v_i \in \acute{V}, v_m \notin \acute{V}, v_m \in V. \end{gathered} \tag{R 2.10}$$

Zvolíme náhodně výchozí vrchol $v_x \in V$ a vytvoříme elementární řešení: Přidáme okruh pouze pro výchozí vrchol, tzn. $T = \{v_x \to v_x\}$. Přidáme v_x do seznamu použitých vrcholů $\mathcal{A}$. V našem příkladě z paragrafu 2.1.1.2.2 elementární řešení konstruujeme takto:



Obr. 2.10 Elementární řešení vkládacích metod v ilustračním příkladě úlohy obchodního cestujícího

Další postup je již specifický pro každou ze zmiňovaných metod.

2.1.2.3.1 Metoda nejbližšího přídavku

Metoda nejbližšího přídavku vyhledá takový vrchol $v_i \in V$, $v_i \notin \mathcal{A}$, že $(v_i; T)$ je minimální. Předpokládejme, že se jedná o vrchol $v_j \in \mathcal{A}$, který je nejblíže vrcholu $v_i \in V$, tzn. $(v_i; T) = h\left((v_i; v_j)\right)$. Dále přepokládejme, že z vrcholu v_j vystupuje trasa $v_j \to v_k$, kde $v_k \in \mathcal{A}$.

Z okruhu T odebereme trasu $v_j \to v_k$ a přidáme trasy $v_j \to v_i$ a $v_i \to v_k$. Nakonec přidáme vrchol v_i do seznamu použitých vrcholů $\mathcal{A}$.

Detailní popis výpočtu příkladu z paragrafu 2.1.1.1.2 zde neuvádíme, jen výsledek pro porovnání s ostatními metodami. Metoda nejbližšího přídavku vydá posloupnost vrcholů v okruhu $\{1,2,3,4,5\}$ a cena tohoto okruhu je 47.

Díky výpočtu příkladu v paragrafu 2.1.1.1.2 také víme, že se jedná o optimální řešení úlohy obchodního cestujícího.

2.1.2.3.2 Metoda nejbližšího vložení

Tato metoda je drobnou modifikací metody nejbližšího přídavku. Výběr přidávaného vrcholu probíhá stejně, modifikace se projeví až při napojení vrcholu.

Mějme vrchol $v_i \in V$, $v_i \notin \mathcal{A}$, takový že $(v_i; T)$ je minimální. Předpokládejme, že je to vrchol $v_j \in \mathcal{A}$, který je nejblíže vrcholu $v_i \in V$, tzn. $(v_i; T) = h\left((v_i; v_j)\right)$. Nyní hledáme v okruhu T trasu $v_k \to v_l$ takovou, aby pro všechny trasy $v_m \to v_n$ v okruhu T byla splněna podmínka

$$h\big((v_k; v_i)\big) + h\big((v_i; v_l)\big) - h\big((v_k; v_l)\big) \leq h\big((v_m; v_i)\big) + h\big((v_i; v_n)\big) - h\big((v_m; v_n)\big). \tag{R 2.11}$$

Vrchol v_k zařadíme mezi vrcholy v_k a v_l, tzn. odebereme z okruhu T trasu $v_k \to v_l$ a přidáme trasy $v_k \to v_i$ a $v_i \to v_l$. Nakonec přidáme vrchol v_i do seznamu použitých vrcholů $\mathcal{A}$.

Detailní popis výpočtu příkladu z paragrafu 2.1.1.1.2 zde neuvádíme, pouze výsledek k porovnání s ostatními metodami. Metoda nejbližšího vložení vydá posloupnost vrcholů v okruhu $\{1,2,3,4,5\}$ a cena tohoto okruhu je 47.

Díky výpočtu příkladu z paragrafu 2.1.1.1.2 také víme, že se jedná o optimální řešení úlohy obchodního cestujícího.

2.1.2.3.3 Metoda nejlevnějšího vložení

Tato metoda je oproti předchozím metodám z paragrafu (2.1.2.3) nejefektivnější, ale implementačně nejnáročnější.

V okruhu T je třeba najít trasu $v_i \to v_j$ a vrchol $v_k \in V$, $v_k \notin \mathcal{A}$ takový, aby pro všechny ostatní trasy $v_m \to v_n$ v okruhu T byla splněna podmínka

$$h\big((v_i; v_k)\big) + h\Big(\big(v_k; v_j\big)\Big) - h\Big(\big(v_i; v_j\big)\Big) \leq h\big((v_m; v_k)\big) + h\big((v_k; v_n)\big) - h\big((v_m; v_n)\big) \quad \text{(R 2.12)}$$

Vrchol v_k zařadíme mezi vrcholy v_i a v_j, tzn. odebereme trasu $v_i \to v_j$ z okruhu T a přidáme trasy $v_i \to v_k$ a $v_k \to v_j$. Nakonec přidáme vrchol v_k do seznamu použitých vrcholů $\mathcal{A}$.

Detailní popis výpočtu příkladu z paragrafu 2.1.1.1.2 zde neuvádíme, pouze výsledek k porovnání s ostatními metodami. Metoda nejlevnějšího vložení vydá posloupnost vrcholů v okruhu $\{1,2,3,4,5\}$ a cena tohoto okruhu je 47.

Díky výpočtu příkladu z paragrafu 2.1.1.1.2 také víme, že se jedná o optimální řešení úlohy obchodního cestujícího.

2.1.2.3.4 Obecný algoritmus vkládacích metod

Při popisu algoritmu se budeme držet značení použitého výše.

I. Zvolíme výchozí vrchol $v_x \in V$ a vytvoříme elementární řešení, $T = \{v_x \to v_x\}$, $\mathcal{A} = \{v_x\}$ (viz Obr. 2.10).
II. Dle vybrané metody pro přidání vrcholu určíme vrchol $v_k \in V$, $v_k \notin \mathcal{A}$ a hranu $v_i \to v_j \in T$.
III. $T \coloneqq T \setminus \{v_i \to v_j\} \cup \{v_i \to v_k, v_k \to v_j\}$.
IV. $\mathcal{A} \coloneqq \mathcal{A} \cup \{v_k\}$. Pokud $\mathcal{A} \neq V$, přejdeme na bod II, jinak skončíme a prohlásíme okruh T za výsledné řešení.

2.1.2.4 *Metoda nejbližšího souseda*

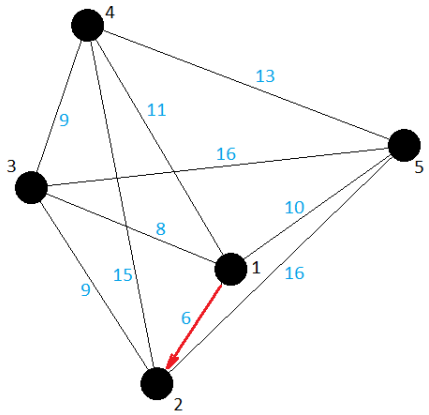
Tato metoda je považována za jednu z nejjednodušších metod pro řešení úlohy obchodního cestujícího. V zahraniční literatuře ji lze najít pod názvem „Nearest Neighbour method“ [20]. Lze ji použít i pro víceokruhové a nesymetrické varianty úlohy obchodního cestujícího, nicméně na rozdíl od jiných metod zde není možné určit odhad odchylky daného řešení od optimálního řešení úlohy obchodního cestujícího.

Uvažujme problém jako graf $G = (V; E)$, kde $V = \{v_1, v_2, \dots, v_N\}$ jsou destinace, $N \in \mathbb{N}$ je počet destinací a E hrany mezi nimi. Ohodnocení hran $H: E \to \mathbb{R}_0^+$ budeme značit jako $h\left((v_i; v_j)\right)$ pro hranu $(v_i; v_j) \in E$. Při popisu budeme značit trasu z v_i do v_j jako $v_i \to v_j$ a zavedeme $\mathcal{A}$ jako pomocný seznam použitých vrcholů. Okruh bude značit posloupnost tras: $T = \{v_m \to v_n, \dots\}$. Na počátku volíme $\mathcal{A} = \emptyset$.

Nejprve zvolíme náhodně výchozí destinaci $v_x \in V$, přidáme ji do seznamu použitých vrcholů $\mathcal{A}$ a poté určíme destinaci $v_i \in V, v_i \notin \mathcal{A}$ s nejnižším ohodnocením tak, aby platil vztah

$$\bigvee_{v_j \in V, v_j \notin \mathcal{A}} \left(h\left((v_x; v_i)\right) \leq h\left((v_x; v_j)\right) \right). \qquad \text{(R 2.13)}$$

Přidáme v_i do seznamu použitých vrcholů $\mathcal{A}$ a přidáme do okruhu T trasu $v_x \to v_i$. Pro náš příklad z paragrafu 2.1.1.1.2 by situace vypadala následovně:



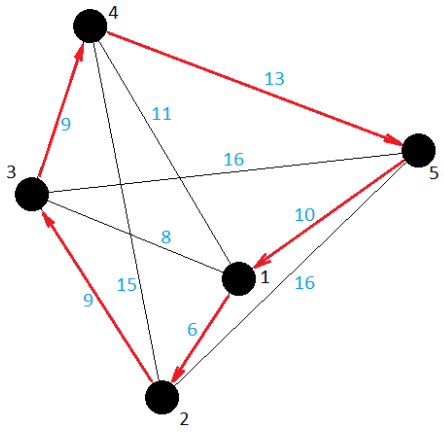
Obr. 2.11 První iterace metody nejbližšího souseda (pro výchozí vrchol 1) v ilustračním příkladě úlohy obchodního cestujícího

Je zřejmé, že destinace v_i je v rámci ohodnocení nejbližší nenavštívenou destinací k destinaci výchozí. V každém dalším kroku algoritmu hledáme nejbližší nenavštívenou destinaci k destinaci předchozí a přidáme ji do okruhu. Označíme-li předchozí destinaci jako $v_j \in V, v_j \in \mathcal{A}$, pak následující destinace musí splňovat

$$\mathrm{V}_{v_k \in V, v_k \notin \mathcal{A}} \left(h\left((v_i; v_j) \right) \leq h\big((v_i; v_k) \big) \right). \qquad \text{(R 2.14)}$$

Iterujeme, dokud nejsou použity všechny vrcholy, tj. $\mathcal{A} = V$.

Předpokládejme, že poslední vrchol, který přidáme do okruhu, je $v_u \in V$. Na závěr přidáme do okruhu T trasu $v_u \rightarrow v_x$, abychom dokončili cyklus (viz Obr. 2.12).



Obr. 2.12 Výsledné řešení dle metody nejbližšího souseda (pro výchozí vrchol 1) v ilustračním příkladě úlohy obchodního cestujícího

Detailní popis výpočtu příkladu z paragrafu 2.1.1.1.2 zde neuvádíme, pouze výsledek k porovnání s ostatními metodami. Metoda nejlevnějšího vložení vydá posloupnost vrcholů v okruhu $\{1,2,3,4,5\}$ a cena tohoto okruhu je 47.

Díky výpočtu příkladu z paragrafu 2.1.1.1.2 také víme, že se jedná o optimální řešení úlohy obchodního cestujícího.

2.1.2.4.1 Popis algoritmu

Při popisu se budeme držet značení definovaného výše.

I. Náhodně zvolíme výchozí vrchol $v_x \in V$, $\mathcal{A} = \{v_x\}$.

II. $(v_i \in V, v_i \notin \mathcal{A}) : \bigvee_{v_j \in V, v_j \notin \mathcal{A}} \left(h\big((v_x; v_i)\big) \leq h\left((v_x; v_j)\right)\right)$.

III. $T \coloneqq T \cup \{v_x \rightarrow v_i\}$, $\mathcal{A} \coloneqq \mathcal{A} \cup \{v_i\}$.

IV. Pokud $\mathcal{A} = V$, pak přejdeme na krok VI, jinak označme poslední přidanou destinaci do okruhu jako $v_i \in V$ a poté

$\big(v_j \in V, v_j \notin \mathcal{A}\big) : \bigvee_{v_k \in V, v_k \notin \mathcal{A}} \left(h\left((v_i; v_j)\right) \leq h\big((v_i; v_k)\big)\right)$.

V. $T \coloneqq T \cup \{v_i \rightarrow v_j\}$, $\mathcal{A} \coloneqq \mathcal{A} \cup \{v_j\}$. Pokud $\mathcal{A} \neq V$ přejdeme na krok IV, jinak přejdeme na krok VI.

VI. Předpokládejme, že poslední vrchol, který jsme přidali do okruhu, je $v_u \in V$, $T \coloneqq T \cup \{v_u \rightarrow v_x\}$.

2.1.2.5 *Simplexová metoda*

Jednou z nejstarších, ale zároveň nejpoužívanějších metod pro řešení úloh lineárního programování je simplexová metoda, která byla poprvé popsána již v roce 1951 Georgem Dantzigem [2].

Simplexovou metodu lze sice také aplikovat na matematický model úlohy obchodního cestujícího z paragrafu 1.4.1, nicméně v současnosti se z důvodu časové náročnosti preferují heuristické algoritmy. Ty sice nemusí nalézt optimální řešení, nicméně jsou mnohem rychlejší.

Vzhledem k tomu, že reálné problémy jsou mnohem náročnější právě z hlediska rychlého rozhodování a časové efektivity, je simplexová metoda v tomto případě v podstatě nevhodná.

Simplexová metoda je obecně známá, proto byl popis této metody vynechán. Popis simplexové metody lze nalézt např. v [21].

2.1.2.6 *Ant Colony Optimization*

Optimalizace mravenčích kolonií (v originále Ant Colony Optimization, dále jen ACO) je soubor meta-heuristických technik používaných v oboru umělé inteligence pro hledání přibližných řešení kombinatorických problémů. Technika se inspiruje chováním mravenců při

hledání nejkratší cesty k potravě, proto ji řadíme mezi další metody využívající inteligence hejna.

Jejich historii lze datovat již od roku 1959, kdy Pierre-Paul Grassé popsal nepřímé předávání informací mravenců pomocí feromonů při budování mraveniště [22]. Základní myšlenku ovšem popsal až M. Dorigo a L. M. Gambardella v roce 1997 [23].

Při hledání potravy mravenci volí svou cestu na základě feromonů ostatních mravenců, jinak čistě náhodně. To znamená, že pokud se jedinec dostane do bodu, ze kterého vede více postupových tras a nemá žádnou feromonovou informaci, pak volí cestu náhodně. Za předpokladu stejné rychlosti všech mravenců bude jedinec A, který našel potravu a zvolil kratší cestu, u potravy dříve než jedinec B, který zvolil cestu delší. Jedinci při návratu do mraveniště vypouští feromonovou informaci, která je důležitá při rozhodování ostatních jedinců jakou trasu zvolit, přičemž jedinec A předá svou informaci rychleji. Při stejném chování všech mravenců se stává feromonová informace o kratší cestě po určité době velmi výrazná a informace o delší cestě následně prakticky zaniká.

V teorii grafů lze mravence simulovat pomocí tzv. „agentů“, přičemž mraveniště a potrava jsou charakterizovány startovním a cílovým uzlem grafu. Jednotlivé souvislé úseky cesty představují hrany grafu a místa, kde lze volit následný postup, jsou uzly grafu. Každá hrana si uchovává informaci v podobě množství feromonu a délky hrany. Pravděpodobnost, že v uzlu $v \in V$ bude vybrána postupová hrana $e \in E$ je vyjádřena pravděpodobnostní přechodovou funkcí takto:

$$P(e) = \frac{\tau(e) \cdot \eta(e)}{\sum_{\acute{e} \epsilon \acute{E}} \tau(\acute{e}) \cdot \eta(\acute{e})}, \qquad \text{(R 2.15)}$$

kde $\tau(e)$ je množství feromonu pro hranu $e \in E$, $\eta(e)$ je převrácená hodnota délky hrany e a $\acute{E}$ je množina hran vedoucích z uzlu $v \in V$.

Aktualizací feromonové informace měníme hodnotu pravděpodobnostní přechodové funkce, a tudíž vhodně zvolená funkce pro aktualizaci feromonové informace může zajistit rychlou konvergenci grafu do stabilního stavu, kdy se hodnoty přechodových funkcí pro nevyužívané hrany budou blížit nule a naopak hrany na analyzované nejkratší trase budou mít vysokou hodnotu pravděpodobnostní přechodové funkce. V závislosti na zvolené funkci

pro aktualizaci feromonové informace rozlišujeme mnoho variant ACO. Zde uvádíme obecný vzorec [24]

$$\tau_{t+1}(e) \coloneqq \begin{cases} (1-p)\cdot\tau_t(e), & pokud\ e \in E\ nebyla\ využita, \\ (1-p)\cdot\tau_t(e)+\varepsilon\,, & jinak, \end{cases} \qquad \text{(R 2.16)}$$

kde $\tau_t \in \mathbb{R}^+$ je feromonová informace v iteraci $t \in \mathbb{N}$, $\varepsilon \in \mathbb{R}^+$je aktualizace feromonové informace a $p \in \langle 0;1\rangle$ tzv. „hodnota vypařování“, v originále „evaporation rate“ [24]. Lze nahlédnout, že rychlost konvergence je závislá především na hodnotě p, přičemž nízká hodnota p má za následek obecně pomalejší konvergenci a naopak. Hodnotu aktualizace feromonové informace ε lze nastavit jako podíl konstanty maxima feromonové informace a délky vysledované nejkratší trasy [24].

Metoda byla upravena i pro rozšířený problém obchodního cestujícího. Každý agent si pamatuje uzly, které navštívil, přičemž agent k v uzlu $i \in V$ se rozhoduje dle pravděpodobnostní přechodové funkce:

$$P(e=(i,j)\in E) = \begin{cases} \dfrac{[\tau(e)]^\alpha \cdot [\eta(e)]^\beta}{\sum_{\acute{e}\in\acute{E}}[\tau(\acute{e})]^\alpha \cdot [\eta(\acute{e})]^\beta}, & pokud\ j \notin M_k, \\ 0, & jinak, \end{cases} \qquad \text{(R 2.17)}$$

kde M_k je množina uzlů, které agent k navštívil, $\alpha, \beta \in \mathbb{R}^+$ jsou konstanty regulující aktualizaci feromonové informace (většinou jsou rovny 1) a $\acute{E}$ je množina hran vedoucích z uzlu i.

V současné době vzniklo mnoho algoritmů založených na této technice, které se liší pravděpodobnostní přechodovou funkcí, případně aktualizací feromonové informace. Uvádíme pouze některé z nich, více lze nalézt např. v[23] [24] [25] a hlavně v [26]:

- **Elitist Ant System**: Aktualizuje feromonovou informaci nejen globálně po dokončení trasy všech jedinců, ale i při dokončení trasy každého jedince.
- **Max-Min Ant System**: V algoritmu je definována maximální a minimální hodnota feromonové informace (τ_{min}, τ_{max}). Feromonová informace je aktualizována pouze v případě, kdy jedinec dosáhne globálního nebo iteračního optima pro daný výpočet. Výchozí hodnota feromonové informace je τ_{max} a je reinicializována pokud dosáhne τ_{min}.

- **Rank-based Ant System (ASrank):** Aktualizace feromonu závisí na délce dosaženého nejlepšího řešení (kratší trasa předává více feromonu).

2.1.2.6.1 Algoritmus

Obecný algoritmus popsal ve své dizertační práci M. Dorigo v roce 1992 [25] a jeho modifikace pro rozšířenou úlohu obchodního cestujícího je uvedena např. v [26].

```
/* Inicializujeme hodnoty každé hrany na výchozí hodnotu.*/
for each e ∈ E:
    τ(e) = τ0;
    η(e) = η0;
end;
place each ant on a randomly chosen node;
for i = 1 to MaxIterations:
    /* Každý jedinec absolvuje svou trasu.*/
    for each ant k do:
        do
            move to the next node according the P(e);
            add this node to the tabu list Mk;
        until the ant hasn't completed the tour;
    end;
    /* Aktualizujeme feromonovou informaci a globální řešení.*/
    for each ant with a complete tour do:
        apply the pheromon update;
        if (ant k's tour is shorter than the global solution)
            update global solution to ant k's tour;
```

2.1.2.7 *Patching method*

Patching method navrhnul již v roce 1979 R. M. Karp [27]. Od té doby se tato metoda dočkala řady modifikací a především v poslední době je analýze a výzkumu této metody věnována velká pozornost [28][29] [30].

Samotná metoda předpokládá znalost optimálního řešení zjednodušeného (tzv. kanonického) dopravního problému (dále jen KDP). Dle značení v paragrafu 1.4.3 pro KDP platí, že $D_i = 1$ a $O_j = 1$, $i = \{1,2,\dots,k\}$, $j = \{1,2,\dots,l\}$.

Předpokládejme, že $N \in \mathbb{N}$ je počet destinací a matice $C_{ij} \in \mathbb{R}_0^+$, $i \neq j$, $i,j \in \{1,2,\dots,N\}$ charakterizuje nákladovou vzdálenost mezi destinacemi.

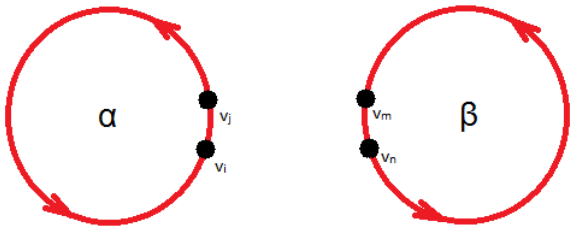
Mějme KDP, kde $D_i = 1$ a $O_j = 1$, $i = \{1,2,\dots,N\}$, $j = \{1,2,\dots,N\}$. Pokud v optimálním řešení KDP vznikne cyklus přes všechny destinace, pak je to řešení úlohy obchodního cestujícího. Jestliže vznikne více cyklů, patching method popisuje techniku spojování cyklů.

Trasu z $v_i \in V$ do $v_j \in V$ budeme značit jako $v_i \to v_j$, $i,j \in \{1,2,\dots,N\}$. Předpokládejme, že existují množiny vrcholů $\mathcal{A}, \mathcal{B} \subseteq V$, $\mathcal{A} \cap \mathcal{B} = \emptyset$ takové, že vrcholy z množiny $\mathcal{A}$ i $\mathcal{B}$ tvoří v optimálním řešení KDP cyklus. Nalezneme takové $v_i \in \mathcal{A}$ a $v_m \in \mathcal{B}$, pro které platí

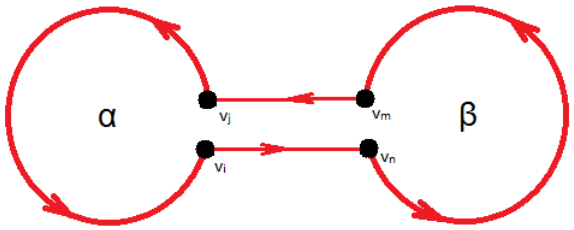
$$C_{im} + C_{nj} - C_{ij} - C_{mn} \leq C_{ko} + C_{pl} - C_{kl} - C_{op}, \qquad \text{(R 2.18)}$$

kde $v_i, v_k \in \mathcal{A}$, $v_m, v_o \in \mathcal{B}$, $v_j \in \mathcal{A}$ (resp. $v_l \in \mathcal{A}$) je vrchol přiřazený v optimálním řešení KDP vrcholu $v_i \in \mathcal{A}$ (resp. $v_k \in \mathcal{A}$) a $v_n \in \mathcal{B}$ (resp. $v_p \in \mathcal{B}$) je vrchol přiřazený v optimálním řešení KDP vrcholu $v_m \in \mathcal{B}$ (resp. $v_o \in \mathcal{B}$).

Nyní provedeme operaci přepojení cyklů. Odstraníme z optimálního řešení KDP trasy $v_i \to v_j$ a $v_m \to v_n$ a přidáme trasy $v_i \to v_n$ a $v_m \to v_j$. Grafická podoba přepojení okruhů je znázorněna na Obr. 2.13 a Obr. 2.14.



Obr. 2.13 - Spojování cyklů v patching method, fáze 1



Obr. 2.14 - Spojování cyklů v patching method, fáze 2

Opakujeme postup přepojování okruhů, dokud nezbývá pouze jeden. Toto řešení prohlásíme za konečné řešení úlohy obchodního cestujícího.

Karp dokázal, že složitost algoritmu je $O(N^3)$[27].

2.1.2.7.1 Popis algoritmu

Při popisu algoritmus se budeme držet značení popsaného výše. Předpokládejme, že v optimálním řešení KDP pro matici C_{ij} vzniklo $d \in \{2, \dots, N\}$ cyklů. Mějme $\mathcal{M} = \{m_1, m_2, \dots, m_d\}$, kde $m_i \in \{1,2, \dots, N\}$ je počet destinací v i-tém cyklu, $i \in \{1,2, \dots, d\}$. J-tou destinaci v i-tém cyklu budeme značit $\mathfrak{M}_{ij} \in V,\ i \in \{1,2, \dots, d\}, j \in \{1,2, \dots, N\}$. Bez újmy na obecnosti budeme předpokládat, že $m_i \leq m_j$ pro $i \leq j$ a že v optimálním řešení vede trasa mezi destinacemi $\mathfrak{M}_{ij}$ a $\mathfrak{M}_{i((j+1) mod m_i)}$ pro $i \in \{1,2, \dots, d\}, j \in \{1,2, \dots, m_i\}$. Trasu mezi destinacemi $v_i \in V$ a $v_j \in V$ budeme značit $v_i \to v_j$. Množinu tras v i-tém cyklu optimálního řešení KDP budeme značit $\mathfrak{L}_i$, $i \in \{1,2, \dots, d\}$. Konečně ohodnocení trasy mezi destinacemi $v_i \in V$ a $v_j \in V$ budeme také značit $C(v_i, v_j)$ (pozn. $C(v_i, v_j) = C_{ij}$).

Výsledný cyklus budeme značit T, na začátku $T = \mathfrak{L}_1$.

I. $(i \in \{1,2,\dots,m_1\}, j \in \{1,2,\dots,m_2\}): \Big(\forall(m \in \{1,2,\dots,m_1\}, n \in \{1,2,\dots,m_2\}): \Big(C\left(\mathfrak{M}_{1i}, \mathfrak{M}_{2((j+1)mod m_2)}\right) + C\left(\mathfrak{M}_{2j}, \mathfrak{M}_{1((i+1)mod m_1)}\right) - C\left(\mathfrak{M}_{1i}, \mathfrak{M}_{1((i+1)mod m_1)}\right) - C\left(\mathfrak{M}_{2j}, \mathfrak{M}_{2((j+1)mod m_2)}\right) \leq C\left(\mathfrak{M}_{1m}, \mathfrak{M}_{2((n+1)mod m_2)}\right) + C\left(\mathfrak{M}_{2j}, \mathfrak{M}_{1((m+1)mod m_1)}\right) - C\left(\mathfrak{M}_{1m}, \mathfrak{M}_{1((m+1)mod m_1)}\right) - C\left(\mathfrak{M}_{2n}, \mathfrak{M}_{2((n+1)mod m_2)}\right)\Big)\Big)$.

II. $T := T \cup \mathfrak{L}_2 \setminus \mathfrak{M}_{1i} \to \mathfrak{M}_{2((j+1)mod m_2)} \cup \mathfrak{M}_{2j} \to \mathfrak{M}_{1((i+1)mod m_1)} \cup \mathfrak{M}_{1i} \to \mathfrak{M}_{1((i+1)mod m_1)} \setminus \mathfrak{M}_{2j} \to \mathfrak{M}_{2((j+1)mod m_2)}; d := d - 1; m_1 := m_1 \cup m_2; \mathcal{M} := \mathcal{M} \setminus m_2;\ update\ \mathfrak{M}_{1i}, i \in \{1,2,\dots,m_1\}; update\ \mathfrak{L}_2;$

III. Pokud $d = 1$, pak skončíme, jinak pokračujeme bodem I.

Výsledná trasa T obsahuje jediný cyklus.

2.2 Metody řešení dopravního problému

V následujícím paragrafu uvedeme několik nejznámějších metod pro řešení dopravního problému, jež byl zformulován v paragrafu 1.4.3. Jedná se především o Vogelovu aproximační metodu, metodu Habrových frekvencí, indexovou metodu a v neposlední řadě i upravenou simplexovou metodu pro klasický dopravní problém [2].

Následující metody hledají řešení pomocí tabulky distribučních nákladů. Tu zkonstruujeme tak, že jako řádky uvedeme dodavatele a jako sloupce odběratele. Každá buňka tabulky odpovídá distribučním nákladům $C_{ij} \in \mathbb{R}_0^+$ pro přepravu jednotky zboží mezi i-tým dodavatelem a j-tým odběratelem.

2.2.1 *Vogelova aproximační metoda*

Vogelova aproximační metoda (metoda VAM) dává řešení „blízké“ optima, proto patří k nejpoužívanějším aproximačním metodám [31]. Metoda je založena na obsazování políček v tabulce distribučních nákladů mezi jednotlivými odběrateli a dodavateli. Jednotlivá políčka charakterizují množství zboží převezeného mezi daným dodavatelem a odběratelem. Řešení

je vyhledáno nejen dle nejvýhodnější sazby nákladů mezi dodavatelem a spotřebitelem, ale i dle rozdílu dvou nejvýhodnějších sazeb pro daného dodavatele. Tím je zajištěno obsazování výhodných spojů v průběhu celého výpočtu rovnoměrně [31].

V každém řádku i sloupci vypočteme rozdíl (neboli diferenci) mezi dvěma nejvýhodnějšími sazbami nákladů pro převoz zboží mezi daným dodavatelem a odběratelem. Nejvýhodnější chápeme jako nejnižší ve smyslu minimalizace nákladů. V každé iteraci algoritmu vybereme řádek s největší diferencí a v tomto řádku hledáme sloupec s nejnižší sazbou nákladů. Políčko tabulky v tomto řádku a sloupci obsadíme maximálním množstvím zboží, které lze přepravit (tj. do výše kapacity dodavatele nebo odběratele). V případě, že uspokojíme poptávku odběratele, vyškrtneme jemu příslušný řádek a přepočítáme řádkové diference. Naopak, pokud uspokojíme nabídku dodavatele, vyškrtneme jemu příslušný sloupec z tabulky a přepočítáme sloupcové diference.

Postup opakujeme ve zmenšené tabulce do té doby, než vyčerpáme kapacity všech dodavatelů a uspokojíme poptávku všech spotřebitelů. Jestliže byl v poslední iteraci vyškrtnut řádek, pak vybíráme sloupec s nejvyšší diferencí a v tomto sloupci položku s nejnižší sazbou. Pokud byl vyškrtnut sloupec, vybíráme řádek s nejvyšší diferencí a v něm položku s nejnižší sazbou.

Vyskytnou-li se současně stejné nejvyšší diference u více řádků, příp. sloupců, obsazujeme přednostně buňku s nižší sazbou distribučních nákladů.

Jsou-li v řádku, příp. v sloupci dvě nebo více shodných nejnižších hodnot distribučních nákladů, pak počítáme diferenci mezi nejvýhodnější a druhou nejvýhodnější sazbou (tak, aby diference nebyla nulová).

2.2.1.1 ***Popis algoritmu:***

Předpokládejme, že $k \in \mathbb{N}$ je počet dodavatelů a $l \in \mathbb{N}$ je počet odběratelů. Množství převáženého zboží mezi i-tým dodavatelem a j-tým odběratelem budeme značit $B_{ij} \in \mathbb{R}^+$, distribuční náklady mezi i-tým dodavatelem a j-tým odběratelem budeme značit $C_{ij} \in \mathbb{R}^+$ a konečně kapacitu i-tého dodavatele jako $D_i \in \mathbb{R}^+$ a poptávku j-tého odběratele jako $O_j \in \mathbb{R}^+$, $i = \{1,2,\dots,k\}, j = \{1,2,\dots,l\}$. Hodnoty B_{ij} jsou na počátku vynulovány.

I. Vypočteme $\mathcal{X}_i$ a $\mathcal{Y}_j$, kde $\mathcal{X}_i$ (resp. $\mathcal{Y}_j$) je rozdíl mezi dvěma nejvýhodnějšími sazbami v i-tém řádku (resp. v j-tém sloupci), $i = \{1,2, \dots, k\}$, $j = \{1,2, \dots, l\}$.

II. V řádku s největší diferencí $\mathcal{X}_i$ najdeme buňku s nejmenší sazbou distribučních nákladů C_{ij} a položíme $B_{ij} = \min(D_i, O_j)$.

III. Pokud $D_i = \sum_{m=1}^{k} B_{mj}$, vyškrtneme i-tý řádek, přepočítáme $\mathcal{X}_i$, $i = \{1,2, \dots, k\}$ i $\mathcal{Y}_j$, $j = \{1,2, \dots, l\}$ a pokračujeme bodem IV. Pokud $O_j = \sum_{n=1}^{l} B_{in}$, vyškrtneme j-tý sloupec, přepočítáme $\mathcal{X}_i$, $i = \{1,2, \dots, k\}$ i $\mathcal{Y}_j$, $j = \{1,2, \dots, l\}$ a pokračujeme bodem V.

IV. Pokud jsou vyškrtnuty všechny sloupce, skončíme. Ve sloupci s největší diferencí $\mathcal{Y}_j$ najdeme buňku s nejmenší sazbou distribučních nákladů C_{ij} a položíme $B_{ij} = \min(D_i - \sum_{m=1,m\neq i}^{k} B_{mj}, O_j - \sum_{n=1,n\neq j}^{l} B_{in})$. Přejdeme na bod III.

V. Pokud jsou vyškrtnuty všechny řádky, skončíme. V řádku s největší diferencí $\mathcal{X}_i$ najdeme buňku s nejmenší sazbou distribučních nákladů C_{ij} a položíme $B_{ij} = \min(D_i - \sum_{m=1,m\neq i}^{k} B_{mj}, O_j - \sum_{n=1,n\neq j}^{l} B_{in})$. Přejdeme na bod III.

2.2.1.2 ***Příklad***

Uvažujme situaci, kdy svážíme brambory ze tří polí do čtyř skladů. Vzdálenosti polí a skladů jsou zadány v kilometrech, produkce polí a kapacity skladů v tunách. Chceme stanovit takový přepravní plán, při kterém ujedou vozidla minimální počet tunokilometrů (tkm). Výchozí podkladové údaje jsou zaznamenány v tabulce Tab. 2.7.

	Sklady, vzdálenosti v km				
Pole	O_1	O_2	O_3	O_4	Předpokládaná produkce polí (v tunách)
D_1	11	11	7	12	50
D_2	6	9	6	10	200
D_3	5	8	11	10	150
Kapacity skladů (v tunách)	120	90	70	120	400

Tab. 2.7 Podkladové údaje pro ilustrační příklad dopravního problému

Úlohu budeme řešit pomocí Vogelovy metody.

Nejprve spočítáme řádkové i sloupcové diference a zapíšeme je do připojeného řádku y_j a sloupce x_i.

	O_1	O_2	O_3	O_4	Předpokládaná produkce polí D_i (v tunách)	x_i
D_1	11 0	11 0	7 0	12 0	50	4
D_2	6 0	9 0	6 0	10 0	200	3
D_3	5 0	8 0	11 0	10 0	150	3
Kapacity skladů O_j (v tunách)	120	90	70	120	400	
y_j	1	1	1	2		

Tab. 2.8 Vogelova metoda v ilustračním příkladě dopravního problému, diference jsou znázorněny červeně, množství převáženého zboží B_{ij} mezi i-tým honem a j-tým skladem zeleně

Největší diference 4 je v prvním řádku. V tomto řádku najdeme nejmenší sazbu, tj. $C_{13} = 7$. Hodnotu B_{13} nastavíme jako $\min\left(D_1 - \sum_{n=1,n\neq 3}^{4} B_{1n}, O_3 - \sum_{m=2}^{3} B_{m3}\right) = 50$. Tím vyčerpáme kapacitu prvního dodavatele D_1, tudíž vyškrtneme první řádek z tabulky.

	O_1	O_2	O_3	O_4	Předpokládaná produkce polí D_i (v tunách)	x_i
~~D_1~~	~~1~~ 0	~~11~~ 0	~~7~~ 50	~~12~~ 0	~~50~~	
D_2	6 0	9 0	6 0	10 0	200	3
D_3	5 0	8 0	11 0	10 0	150	3
Kapacity skladů O_j (v tunách)	120	90	70	120	400	
y_j	1	1	1	2		

Tab. 2.9 Vogelova metoda v ilustračním příkladě dopravního problému po první iteraci

Po vyškrtnutí řádku přepočítáme sloupcové diference ve zmenšené tabulce. Největší diference je 5 v třetím sloupci. V tomto sloupci najdeme nejmenší sazbu, tj. $C_{23} = 6$.

Hodnotu B_{23} nastavíme jako $\min\left(D_2 - \sum_{n=1,n\neq 2}^{4} B_{2n}\,, O_3 - \sum_{m=1,m\neq 2}^{3} B_{m3}\right) = 20$. Tímto vyčerpáme kapacitu třetího skladu O_3, tudíž vyškrtneme třetí sloupec z tabulky.

	O_1	O_2	~~O_3~~	O_4	Předpokládaná produkce polí D_i (v tunách)	x_i
~~D_1~~	~~11~~ 0	~~11~~ 0	~~7~~ 50	~~12~~ 0	~~50~~	
D_2	6 0	9 0	~~6~~ 20	10 0	200	3
D_3	5 0	8 0	~~11~~ 0	10 0	150	3
Kapacity skladů O_j (v tunách)	120	90	~~70~~	120	400	
y_j	1	1		0		

Tab. 2.10 Vogelova metoda v ilustračním příkladě dopravního problému po druhé iteraci

Po vy škrtnutí sloupce přepočítáme řádkové diference. Nyní máme dvě stejně velké diference 3 ve druhém a třetím řádku. V těchto řádcích vyhledáme nejmenší sazbu, tj. $C_{31} = 5$. Hodnotu B_{31} nastavíme jako $\min(D_3 - \sum_{n=2}^{4} B_{3n}\,, O_1 - \sum_{m=1}^{2} B_{m1}) = 120$. Tímto vyčerpáme kapacitu prvního skladu O_3, tudíž vyškrtneme první sloupec z tabulky.

	O_1	O_2	O_3	O_4	Předpokládaná produkce polí D_i (v tunách)	$\mathcal{X}_i$
D_1	11 0	11 0	7 50	12 0	50	
D_2	6 0	9 0	6 20	10 0	200	1
D_3	5 120	8 0	11 0	10 0	150	2
Kapacity skladů O_j (v tunách)	120	90	70	120	400	
$\mathcal{Y}_j$		1		0		

Tab. 2.11 Vogelova metoda v ilustračním příkladě dopravního problému po třetí iteraci

Po vyškrtnutí sloupce přepočítáme řádkové diference. Největší diferenci 2 obsahuje třetí řádek. V tomto řádku najdeme nejnižší sazbu, tj. $C_{32} = 8$. Hodnotu B_{32} nastavíme jako $\min\left(D_3 - \sum_{n=1,n\neq 2}^{4} B_{3n}\,, O_2 - \sum_{m=1}^{2} B_{m2}\right) = 30$. Tímto vyčerpáme kapacitu třetího dodavatele D_3, tudíž vyškrtneme třetí řádek z tabulky.

	O_1	O_2	O_3	O_4	Předpokládaná produkce polí D_i (v tunách)	$\mathcal{X}_i$
D_1	11 0	11 0	7 50	12 0	50	
D_2	6 0	9 0	6 20	10 0	200	1
D_3	5 120	8 30	11 0	10 0	150	
Kapacity skladů O_j (v tunách)	120	90	70	120	400	
$\mathcal{Y}_j$		0		0		

Tab. 2.12 Vogelova metoda v ilustračním příkladě dopravního problému po čtvrté iteraci

Pro zbývající hodnoty D_2O_2 a D_2O_4 nelze počítat diference, proto je obsadíme postupně v pořadí od nižší sazby. Nižší sazbu má $C_{22} = 9$, tudíž hodnotu B_{22} nastavíme jako $\min\left(D_2 - \sum_{n=1,n\neq 2}^{4} B_{2n}\,, O_2 - \sum_{m=1,m\neq 2}^{3} B_{m2}\right) = 60$. Zbývající hodnotu B_{24} nastavíme jako $\min\left(D_2 - \sum_{n=1}^{3} B_{2n}\,, O_2 - \sum_{m=1,m\neq 2}^{3} B_{m4}\right) = 120$.

Nyní je uspokojena poptávka odběratelů i nabídka dodavatelů.

	O_1	O_2	O_3	O_4	Předpokládaná produkce polí D_i (v tunách)	$\mathcal{X}_i$
D_1	11 0	11 0	7 50	12 0	50	
D_2	6 0	9 0	6 20	10 120	200	
D_3	5 120	8 30	11 0	10 0	150	
Kapacity skladů O_j (v tunách)	120	90	70	120	400	
$\mathcal{Y}_j$						

Tab. 2.13 Vogelova metoda v ilustračním příkladě dopravního problému v poslední iteraci

Výsledné řešení v tunokilometrech (tkm) spočteme jako $\sum_{i=1}^{3}\sum_{j=1}^{4} B_{ij} \cdot C_{ij} = 3050\ tkm\ (= 7 * 50 + 9 * 60 + 6 * 20 + 10 * 120 + 5 * 120 + 8 * 30)$.

2.2.2 *Metoda Habrových frekvencí*

Metoda Habrových frekvencí je aproximační metoda, která byla zmíněna již v roce 1966 jako metoda výhodná pro optimalizaci v ekonomické praxi [32]. Metoda porovnává sazby nejen v řádku, ale i mezi sloupci a napříč celou tabulkou (získanou hodnotu nazveme „Habrovou frekvencí“), což pro velké rozsahy není efektivní a lze očekávat větší implementační náročnost než u Vogelovy metody.

Metoda se dočkala řady modifikací, především ve výpočtu Habrovy frekvence, která je v původní Habrově definici [32] popsána vztahem

$$F_{ij} = \sum_{m=1}^{k}\sum_{n=1}^{l}\left(C_{ij} + C_{mn} - C_{in} - C_{mj}\right), \quad i = \{1,2,\dots,k\}, j = \{1,2,..,l\}, \tag{R 2.19}$$

kde $k \in \mathbb{N}$ je počet dodavatelů, $l \in \mathbb{N}$ je počet odběratelů a $C_{ij} \in \mathbb{R}^{+}$ jsou distribuční náklady mezi i-tým dodavatelem a j-tým odběratelem.

Výpočet Habrovy frekvence se často modifikuje tak, aby byl výpočet efektivnější:

$$F_{ij} = C_{ij} - \frac{r_i}{k} - \frac{t_j}{l}, \qquad \text{(R 2.20)}$$

kde r_i je součet sazeb v i-tém řádku a t_j je součet sazeb v j-tém sloupci, $i = \{1,2,\dots,k\}, j = \{1,2,..,l\},$. Tento vztah lze odvodit ze základního vztahu pro Habrovu frekvenci lineární transformací [12].

2.2.2.1 *Popis algoritmu*

Předpokládejme, že $k \in \mathbb{N}$ je počet dodavatelů a $l \in \mathbb{N}$ je počet odběratelů. Množství převáženého zboží mezi i-tým dodavatelem a j-tým odběratelem budeme značit $B_{ij} \in \mathbb{R}^+$, distribuční náklady mezi i-tým dodavatelem a j-tým odběratelem budeme značit $C_{ij} \in \mathbb{R}^+$ a konečně kapacitu i-tého dodavatele jako $D_i \in \mathbb{R}^+$ a poptávku j-tého odběratele jako $O_j \in \mathbb{R}^+$, $i = \{1,2,\dots,k\}, j = \{1,2,\dots,l\}$. Hodnoty B_{ij} jsou na počátku vynulovány.

I. Vypočteme Habrovy frekvence dle vztahu $F_{ij} = \sum_{m=1}^{k}\sum_{n=1}^{l}(C_{ij} + C_{mn} - C_{in} - C_{mj}), i = \{1,2,\dots,k\}, j = \{1,2,..,l\}$.
II. Nechť F_{ij} je Habrova frekvence s nejnižší hodnotou. Potom $B_{ij} = \min(D_i - \sum_{m=1,m\neq i}^{k} B_{mj}, O_j - \sum_{n=1,n\neq j}^{l} B_{in})$.
III. Pokud je $O_j = \sum_{m=1}^{k} B_{mj}$, vyškrtneme j-tý sloupec. Pokud je $D_i = \sum_{n=1}^{l} B_{in}$, vyškrtneme i-tý řádek.
IV. Postup opakujeme od bodu II. pro zmenšenou tabulku distribučních nákladů. Pokud $O_j = \sum_{m=1}^{k} B_{mj}$ a $D_i = \sum_{n=1}^{l} B_{in}$, $i = \{1,2,\dots,k\}, j = \{1,2,..,l\}$, pak skončíme.

Detailní výpočet příkladu z paragrafu 2.2.1.2 pomocí metody Habrových frekvencí neuvádíme, pouze výsledek pro srovnání s ostatními metodami. Vypočtená hodnota tunokilometrů je shodná s vypočtenou hodnotou pomocí Vogelovy metody, tj. $3050\ tkm$.

2.2.3 *Indexová metoda*

Indexová metoda patří mezi jednu z nejjednodušších aproximačních metod [31]. Je také založena na obsazování políček v tabulce distribučních nákladů mezi jednotlivými odběrateli a dodavateli. Přednostně jsou obsazována políčka s nižšími náklady.

Políčka obsazujeme postupně od nejvýhodnější (nejmenší) sazby maximálně možným množstvím (tj. množství do výše kapacity dodavatele, příp. odběratele). Při vyčerpání kapacity dodavatele vyškrtneme příslušný řádek, při uspokojení požadavku spotřebitele vyškrtneme příslušný sloupec [31].

Ve zmenšené tabulce distribučních nákladů opět nalezneme nejmenší sazbu a opakujeme postup až do vyčerpání kapacity všech dodavatelů, resp. požadavků všech odběratelů.

Jestliže je nejnižší sazba shodná u více políček, přednostně obsadíme tu sazbu, pro kterou můžeme přepravit větší množství produktu. Políčka s nulovou sazbou (u fiktivního dodavatele nebo odběratele) obsazujeme jako poslední [31].

2.2.3.1 *Popis algoritmu*

Předpokládejme, že $k \in \mathbb{N}$ je počet dodavatelů a $l \in \mathbb{N}$ je počet odběratelů. Množství převáženého zboží mezi i-tým dodavatelem a j-tým odběratelem budeme značit $B_{ij} \in \mathbb{R}^+$, distribuční náklady mezi i-tým dodavatelem a j-tým odběratelem budeme značit $C_{ij} \in \mathbb{R}^+$ a konečně kapacitu i-tého dodavatele jako $D_i \in \mathbb{R}^+$ a poptávku j-tého odběratele jako $O_j \in \mathbb{R}^+$, $i = \{1,2,\dots,k\}$, $j = \{1,2,\dots,l\}$. Hodnoty B_{ij} jsou na počátku vynulovány. Dále zavedeme seznam uvažovaných dodavatelů $\mathcal{X} = \{1,2,\dots,k\}$ a seznam uvažovaných odběratelů $\mathcal{Y} = \{1,2,\dots,l\}$.

I. Hledám takové C_{ij}, pro které platí $C_{ij} \leq C_{kl}, i,k \in \mathcal{X}, j,l \in \mathcal{Y}$. Potom $B_{ij} = \min\left(D_i - \sum_{n=1,n\neq j}^{l} B_{in}, O_j - \sum_{m=1,m\neq i}^{k} B_{mj}\right)$.
II. Pokud $D_i = \sum_{n=1}^{l} B_{in}$, $\mathcal{X} := \mathcal{X} \setminus \{i\}$. Pokud $O_j = \sum_{m=1}^{k} B_{mj}$, $\mathcal{Y} := \mathcal{Y} \setminus \{j\}$.
III. Jestliže $\sum_{i=1}^{k} B_{ij} = O_j$ pro $j \in \{1,2,\dots,l\}$ nebo $\sum_{j=1}^{l} B_{ij} = D_i$ pro $i \in \{1,2,\dots,k\}$, skončíme. Jinak opakujeme bod I.

Detailní výpočet příkladu z paragrafu 2.2.1.2 pomocí indexové metody neuvádíme, pouze výsledek pro srovnání s ostatními metodami. Vypočtená hodnota tunokilometrů je $3100\ tkm$.

2.2.4 *Upravená simplexová metoda pro klasický dopravní problem*

Metoda byla popsána G. Dantzigem v publikaci *„Application of the simplex method to a transportation problem"*. Tato metoda je notoricky známá a proto ji zde neuvádíme.

Podrobný popis metody lze nalézt např. v [21].

2.2.5 *Maďarská metoda*

Maďarská metoda patří mezi další z řad aproximačních metod, nicméně je určena pro řešení zjednodušeného (tzv. kanonického) dopravního problému (dále jen KDP). Dle značení v paragrafu 1.4.3 pro KDP platí, že $D_i = 1$ a $O_j = 1$, $i = \{1,2, \dots, k\}$, $j = \{1,2, \dots, l\}$.

Metoda byla popsána H. Kuhnem v roce 1955 [33] a je známá i pod názvem Kuhnův-Munkresův algoritmus, případně Munkresův přiřazovací algoritmus.

Metoda je založena také na obsazování políček v tabulce distribučních nákladů mezi jednotlivými odběrateli a dodavateli.

Nejprve vybereme v každém řádku minimální sazbu distribučních nákladů a tuto sazbu odečteme od všech hodnot v daném řádku. Následně vybereme v každém sloupci minimální sazbu distribučních nákladů a tuto sazbu odečteme od všech hodnot v daném sloupci. Takto vznikne v každém řádku i sloupci alespoň jedna nulová sazba.

Nyní budeme hledat „silně" a „slabě" nezávislé nulové sazby. Řekneme, že nulová sazba v i-tém řádku a j-tém sloupci je silně nezávislá, pokud se jedná o jedinou nulovou sazbu v i-tém řádku i j-tém sloupci. Jinak se jedná o slabě nezávislou nulovou sazbu.

Vyhledáme všechny silně nezávislé nuly a pro každou silně nezávislou nulu v i-tém řádku a j-tém sloupci vyškrtneme tento sloupec i řádek z matice a dále uvažujeme pouze redukovanou submatici. Následně vybereme stejným způsobem i slabě nezávislé nuly.

Je zřejmé, že tímto řešením získáme k (slabě i silně) nezávislých nulových sazeb. Předpokládejme, že nezávislá nulová sazba je v m-tém řádku a n-tém sloupci. Pak pro řešení

KDP platí, že m-tý dodavatel obsluhuje n-tého odběratele. Jelikož nezávislých nulových sazeb je k, pokryjeme všechny dodavatele.

Uvažovaný příklad z paragrafu 2.2.1.2 nelze touto metodou algoritmem řešit.

Složitost původního algoritmu popsaného Kuhnem je $O(n^4)$.

2.2.5.1 *Popis algoritmu*

Při popisu algoritmu se budeme držet značení definovaného výše. Zavedeme seznam uvažovaných řádků $\alpha \subseteq \{1,2,\dots,k\}$ a seznam uvažovaných sloupců $\beta \subseteq \{1,2,\dots,l\}$. Dále zavedeme pomocnou proměnnou $X_{ij} \in \{0,1\}$, $i \in \{1,2,\dots,k\}$, $j \in \{1,2,\dots,l\}$, která vyjadřuje, zda i-tý dodavatel obsluhuje j-tého odběratele ($X_{ij} = 1$ pokud ano, jinak $X_{ij} = 0$). Na počátku $X_{ij} = 0$, $i \in \{1,2,\dots,k\}$, $j \in \{1,2,\dots,l\}$ a $\alpha = \{1,2,\dots,k\}$, $\beta = \{1,2,\dots,l\}$.

I. $\forall(u \in (1,2,\dots,k)) : (C_{uv} = \min_{k\in\{1,2,\dots,l\}} C_{uk}, \forall(j \in (1,2,\dots,l)) : (C_{uj} \coloneqq C_{uj} - C_{uv}))$.

II. $\forall(v \in (1,2,\dots,l)) : (C_{uv} = \min_{m\in\{1,2,\dots,k\}} C_{mv}, \forall(j \in (1,2,\dots,k)) : (C_{jv} \coloneqq C_{jv} - C_{uv}))$.

III. $\forall\left(C_{ij} = 0, i \in \alpha, j \in \beta, \nexists(m \in \{1,2,\dots,k\}, m \neq i) : (C_{mj} = 0), \nexists(n \in \{1,2,\dots,l\}, n \neq j) : (C_{in} = 0)\right) : (\alpha \coloneqq \alpha \setminus \{i\}, \beta \coloneqq \beta \setminus \{j\}, X_{ij} = 1)$.

IV. $\forall(C_{ij} = 0, i \in \alpha, j \in \beta) : (\alpha \coloneqq \alpha \setminus \{i\}, \beta \coloneqq \beta \setminus \{j\}, X_{ij} = 1)$.

Pro každé $X_{ij} = 1$ i-tý dodavatel obsluhuje j-tého odběratele.

2.3 Metody řešení přiřazovacího problému

Jak již bylo zmíněno, přiřazovací problém je NP-úplný, tudíž pro její řešení v podstatě neexistuje složitostně efektivní algoritmus, který by dával optimální řešení. Při popisu metod se zaměříme především na heuristické metody, neboť jsou časově efektivní a jejich řešení je pro zkoumané účely dostačující.

2.3.1 *Mayerova metoda*

Mayerovu metodu řadíme mezi nejpoužívanější metody pro konstrukci okruhů ve víceokruhovém dopravním problému [12]. Dle paragrafu 1.4.2 cílem metody je rozdělit objednávky vozidlům, přičemž minimalizujeme počet okruhů vozidel. Každé vozidlo je omezeno svou kapacitou a dobou provozu. Zahajuje a končí svůj okruh v depu (základně), ze kterého se distribuuje zboží.

Mějme posloupnost $V = \{v_1, v_2, \dots, v_m\}$, kde $v_i \in \mathbb{R}^+$ značí kapacitu i-tého vozidla, $i \in \{1,2, \dots, m\}$ a $m \in \mathbb{N}$ je počet vozidel. Předpokládejme, že vozidla jsou umístěna v depu (základně) a jejich okruhy začínají i končí v tomto místě. Bez újmy na obecnosti předpokládejme, že vozidla $i \in \{1,2, \dots m\}$ jsou seřazena dle nákladů na provoz vzestupně, kde $m \in \mathbb{N}$ je počet vozidel. Pokud by tomu tak nebylo, vozidla lze přečíslovat. Tento předpoklad zajistí, že ve výsledném řešení víceokruhového dopravního problému budou použita vozidla právě v pořadí $1,2, \dots, m$.

Dále mějme posloupnost objednávek $O \in \{o_1, o_2, \dots, o_n\}$, kde $o_j \in \mathbb{R}^+$ značí kapacitu j-té objednávky, $j \in \{1,2, \dots, n\}$ a $n \in \mathbb{N}$ je počet objednávek. Pro každou objednávku o_i a o_j definujeme $C_i \in \mathbb{R}^+$ jako vzdálenost i-té objednávky od depa (základny) a $C_{ij} \in \mathbb{R}^+$ jako vzdálenost mezi i-tou a j-tou objednávkou. Bez újmy na obecnosti předpokládejme, že objednávky v posloupnosti O jsou seřazeny dle vzdálenosti od depa (základny) sestupně. Pokud by tomu tak nebylo, objednávky lze přečíslovat.

V každé iteraci algoritmu vybíráme první nevyřízenou objednávku o_u z posloupnosti O a přiřadíme ji prvnímu volnému vozidlu j, jehož kapacita je větší než kapacita objednávky ($o_u \leq v_j$), $u \in \{1,2, \dots, n\}$, $j \in \{1,2, \dots, m\}$. Odebereme objednávku o_u z posloupnosti O a dale postupujeme v přiřazování objednávek podobně jako v Jarníkově algoritmu [15]. Hledáme objednávku $o_x \in O$ takovou, jejíž vzdálenost od množiny dosud přiřazených objednávek je minimální. Pokud označíme množinu již přiřazených objednávek jako $\mathcal{A} \subseteq O$, pak lze tuto podmínku vyjádřit takto

$$C_{xj} \leq C_{ij}, \quad o_j \in \mathcal{A}, o_x, o_i \in O \setminus \mathcal{A}. \tag{R 2.21}$$

kde $i, j, x \in \{1,2, \dots, n\}$.

Pokud kapacita vozidla v_f je větší nebo rovna sumě kapacit přiřazených objednávek z množiny $\mathcal{A}$ a kapacitě objednávky o_x, pak přidáme tuto objednávku do množiny $\mathcal{A}$. Iterujeme v přiřazování dalších objednávek dle postupu popsaného v odstavci výše, dokud vozidlo j kapacitně pokrývá objednávky z $\mathcal{A}$.

Vypočteme odhad času, který je potřeba pro obsloužení objednávek z $\mathcal{A}$. Tento odhad lze provést některým z algoritmů pro úlohu obchodního cestujícího, kde místo distinční matice použijeme časovou matici. Pokud tento odhad společně s časovými odhady obsloužení ostatních okruhů přiřazených k vozidlu j překračuje dobu provozu vozidla j, obereme z posloupnosti $\{1,2,\dots,m\}$ vozidlo j, odebereme všechny objednávky z $\mathcal{A}$ a iterujeme celý algoritmus pro první volné vozidlo seznamu $\{1,2,\dots,m\}$. V opačném případě odebereme všechny objednávky $\mathcal{A}$ z posloupnosti O ($O \coloneqq 0 \setminus \mathcal{A}$) a přiřadíme vozidlu j okruh, který se skládá z objednávek z $\mathcal{A}$. Dále iterujeme algoritmus se stejným vozidlem.

Postup opakujeme, dokud existuje objednávka nepřiřazená k žádnému vozidlu.

2.3.1.1 *Popis algoritmu*

Při popisu algoritmu využíváme značení definované výše. Předpokládáme posloupnost V setříděnou dle provozních nákladů vzestupně a posloupnost O setříděnou dle vzdálenosti od základny sestupně. Mějme pomocné proměnné: množinu již přiřazených objednávek $\mathcal{A}$ a vozidlo připravené k přiřazování jako v_1. Na začátku v_1 je první vozidlo z posloupnosti $\{1,2,\dots,m\}$. Dále $\mathcal{A} \coloneqq \{o_1\}$, kde o_1 je první objednávka z posloupnosti O, která je kapacitně menší nebo rovna kapacitě vozidla v_1.

I. Pokud $O = \emptyset$, skončíme. Jinak hledáme (o_x): $\left(C_{xj} \leq C_{ij}, o_j \in \mathcal{A}, o_x, o_i \in O \setminus \mathcal{A}\right)$.

II. Pokud $v_1 \geq \sum_{a \in O} a + o_x$, pak $\mathcal{A} \coloneqq \mathcal{A} \cup \{o_x\}$ a pokračujeme bodem I. Jinak pokračujeme bodem III.

III. Pokud nový okruh dle objednávek $\mathcal{A}$ společně s ostatními okruhy pro vozidlo v_1 splňuje časové omezení provozu vozidla v_1, pak $O \coloneqq O \setminus \mathcal{A}, \mathcal{A} = \{o_1\}$, kde o_1 je první objednávka z posloupnosti O, která je kapacitně menší nebo rovna kapacitě vozidla v_1 a pokračujeme bodem I. Jinak přiřadíme vozidlu v_1 okruh dle

objednávek $\mathcal{A}$, odebereme vozidlo v_1 z posloupnosti $\{1,2,\ldots,m\}$ a označíme v_1 jako první vozidlo z posloupnosti $\{1,2,\ldots,m\}$.

2.3.1.2 *Příklad*

Příklad byl převzat z publikace [12].

Jako testovací úloha byl použit problém rozvozu lístků pro sčítání obyvatel. Centrálním místem je Praha, ostatními místy chápeme 12 krajských měst České republiky. Kapacity míst jsou počty obyvatel jednotlivých krajů v tisících (zaokrouhleno). K dispozici je jedno vozidlo o kapacitě 2100. Vzdálenosti a kapacity jednotlivých měst jsou uvedeny v tabulce:

Místo	Kapacity	Vzdálenosti											
		Os	Zl	Ol	Br	ČB	KV	Ji	HK	Pa	Li	Pl	ÚnL
Pr	-	362	297	285	202	140	133	123	112	104	102	94	92
Os	1278	-	104	93	165	346	495	253	240	240	335	456	454
Zl	598		-	63	100	281	430	188	212	210	309	391	389
Ol	641			-	78	259	408	166	149	147	246	369	367
Br	1136				-	186	335	93	142	138	239	296	294
ČB	626					-	216	126	217	196	242	133	232
KV	304						-	256	245	237	205	83	122
Ji	521							-	110	89	185	186	215
HK	551								-	21	97	206	166
Pa	509									-	118	198	196
Li	429										-	196	92
Pl	551											-	146
ÚnL	827												-

Tab. 2.14 Vzdálenosti a kapacity krajských měst v ilustračním příkladě přiřazovacího problému

Krajská města jsou ve zkratkách označena takto: Praha (Pr), Ostrava (Os), Zlín (Zl), Olomouc (Ol), Brno (Br), České Budějovice (ČB), Karlovy Vary (KV), Jihlava (Ji), Hradec Králové (HK), Pardubice (Pa), Liberec (Li), Plzeň (Pl), Ústí nad Labem (ÚnL).

Dle postupu Mayerovy metody volíme nejvzdálenější destinaci od centrálního místa, což je Ostrava. Postupně přiřazujeme vozidlu další destinace nejblíže vzdálené dosud přiřazeným destinacím. V tomto příkladě se jedná o Olomouc. Tím vyčerpáme kapacitu vozidla, neboť

suma kapacit těchto objednávek je 1919 a každá další objednávka by překročila jeho kapacitu.

Postupujeme v přiřazování objednávek do dalšího okruhu vozidla. Jako nejvzdálenější nepřiřazenou objednávku volíme destinaci Zlín. Nejbližší destinace k destinaci Zlín je Brno. Tím opět vyčerpáme kapacitu vozidla, neboť suma kapacit těchto objednávek je 1734 a nejbližší destinace k množině již přiřazených objednávek je Jihlava, která by svou kapacitou překročila kapacitu vozidla.

Další okruh začne v Českých Budějovicích, nejbližší destinací je poté Jihlava a konečně Pardubice. Tímto opět vyčerpáme kapacitu vozidla, neboť další destinací by byl Hradec Králové, jehož kapacita by překročila kapacitu vozidla.

Další okruh začne v Karlových Varech, nejbližší destinací je Plzeň, poté Ústí nad Labem.

Konečně posledním okruhem je Hradec Králové a Liberec.

Mayerova metoda tedy zkonstruuje 5 okruhů, kde první okruh obsahuje Ostravu a Olomouc, druhý okruh Zlín a Brno, třetí okruh České Budějovice, Jihlavu a Pardubice, čtvrtý okruh Karlovy Vary, Plzeň a Ústí nad Labem a konečně pátý okruh Hradec Králové a Liberec.

2.3.2 *Metoda Fernandez de la Vega-Luecker*

Metoda Fernandez de la Vega-Luecker byla popsána již v roce 1981 dvojicí matematiků F. de la Vegou a G. S. Lueckerem [34]. Metoda řeší přiřazovací problém v lineárním čase [34].

Dle paragrafu 1.4.2 cílem metody je rozdělit objednávky vozidlům, přičemž minimalizujeme počet okruhů vozidel. Metoda předpokládá shodné kapacity všech vozidel. Budeme ji značit $\zeta \in \mathbb{R}^+$.

Hlavní myšlenkou metody je rozdělit objednávky k vozidlům dle kapacit na ty, jejichž přiřazení je optimální a zbytek, které jsou přiřazeny některou z heuristických metod. Toto rozdělení je závislé na vstupním parametru, tudíž složitost metody závisí na tomto parametru.

Metoda pracuje se vstupním parametrem $r \in \langle 0; \infty)$, který udává, o jakou část matematicky optimálního počtu okruhů může být vypočtené řešení horší než skutečné optimum.

Mějme posloupnost $V = \{v_1, v_2, \dots, v_m\}$, kde $v_i \in \mathbb{R}^+$ značí kapacitu i-tého vozidla, $i \in \{1,2, \dots, m\}$ a $m \in \mathbb{N}$ je počet vozidel. Předpokládejme, že vozidla jsou umístěna v depu (základně) a jejich okruhy začínají i končí v tomto místě. Bez újmy na obecnosti předpokládejme, že vozidla $\{1,2, \dots, m\}$ jsou seřazena dle nákladů na provoz vzestupně. Pokud by tomu tak nebylo, vozidla lze přečíslovat. Tento předpoklad zajistí, že ve výsledném řešení víceokruhového dopravního problému budou použita vozidla právě v pořadí, v jakém byla uvedena v posloupnosti $\{1,2, \dots, m\}$.

Dále mějme posloupnost objednávek $O \in \{o_1, o_2, \dots, o_n\}$, kde $o_j \in \mathbb{R}^+$ značí kapacitu j-té objednávky, $j \in \{1,2, \dots, n\}$ a $n \in \mathbb{N}$ je počet objednávek.

Objednávky, jejichž kapacita je větší nebo rovna $\frac{r*\zeta}{2}$ umístíme do $\left\lceil \frac{\sum_{a \in O} a}{\zeta} \right\rceil$ prvních vozidel ze seznamu V některým z algoritmů dávajících optimální řešení přiřazovacího problému.

Zbylé objednávky seřadíme sestupně dle kapacit a přiřazujeme do již využitých vozidel metodou „first-fit" [7], aneb přidáme je do prvního vozidla v seznamu, které je schopno kapacitně pokrýt tuto objednávku.

Metoda Fernandez de la Vega se při aplikaci na praktické problémy ukázala jako výhodnější než Mayerova metoda [12].

Z důvodu velkého rozsahu předložené práce neuvádíme kompletní výpočet příkladu z paragrafu 2.3.1.2, ale uvedeme zde pouze výsledek kvůli porovnání s ostatními metodami. Metoda Fernandez de la Vega-Luecker pro $r = 0{,}5$ zkonstruuje pouze 4 okruhy. První okruh tvoří Ostrava a Olomouc, druhý okruh Zlín, Brno a Karlovy Vary, třetí České Budějovice, Plzeň a Ústí nad Labem, poslední okruh je tvoří Liberec, Hradec Králové, Pardubice a Jihlava.

2.3.2.1 *Popis algoritmu*

Při popisu algoritmu se budeme držet značení popsaného výše.

I. Objednávky, jejichž kapacita je větší nebo rovna $\frac{r*\zeta}{2}$ umístíme do $\left\lceil \frac{\sum_{a \in O} a}{\zeta} \right\rceil$ prvních vozidel ze seznamu některým z algoritmů dávajících optimální řešení přiřazovacího problému [7].

II. Zbylé objednávky seřadíme dle kapacit sestupně.

III. Procházíme postupně zbylé objednávky a přidáváme je do prvního vozidla z $\left\lceil \frac{\sum_{a \in O} a}{\zeta} \right\rceil$ v seznamu, které je schopno kapacitně pokrýt tuto objednávku.

3 Reálný dopravní problém

V následujících paragrafech je popsán nejprve reálný dopravní problém v rozšířené variantě. Dále pak popíšeme vlastní metody řešení pro vybrané varianty tohoto problému. Návrhy řešení se inspirují metodami, které byly studovány v rámci kapitoly 2.

3.1 Model rozšířeného dopravního problému

Následující model je inspirován jedním praktickým dopravním problémem. Předpokládejme, že výrobní podnik má k dispozici vozový park a každý den rozváží vyrobené zboží do stanovených destinací. Ekonomickým cílem podniku je maximalizace zisku a s tím související minimalizace nákladů na rozvoz výrobků.

Při matematické formulaci tohoto problému budeme uvažovat pouze nejdůležitější podmínky, které je třeba splnit. Detailní formulace problému je vzhledem k variabilitě možných restrikcí a podmínek velice náročná a přesahuje rámec této práce.

Předpokládejme, že celkový vozový park je složen z konečně mnoha oblastních vozových parků, které mohou být umístěny obecně na různých místech.

Při definici modelu se budeme držet značení popsaného v rámci paragrafu (1.4).

$N \in \mathbb{N}$, .. Celkový počet vozidel ve všech vozových parcích.

$U \in \mathbb{N}$, .. Počet oblastních vozových parků.

$|K_i| \in \mathbb{N}$, $i \in \{1,2, \dots, U\}$. .. Počet vozidel v i-tém oblastním vozovém parku.

Pro úplnost zavedeme následující podmínky. Pro žádné vozidlo není povoleno, aby bylo evidováno ve dvou vozových parcích zároveň, a sjednocením vozidel ve všech oblastních vozových parcích dostáváme celkový počet vozů.

$$K_i \cap K_j = \emptyset,$$
$$i, j \in \{1,2, \dots, U\}, i \neq j, \tag{R 3.1}$$

$$\bigcup_{m=1}^{U} K_m = K, \tag{R 3.2}$$

kde K_i je množina vozidel v i-tém oblastním vozovém parku, K je množina všech vozidel a $i \in \{1,2, \dots, U\}$.

Pro každé vozidlo definujeme náklady na kilometr provozu, náklady na hodinu provozu, jeho kapacitu a také dodatečné vybavení vozidla. Každé vozidlo je omezeno maximální dobou provozu. Pro $i \in \{1,2,..,U\}$ a $j \in \{1,2,..,K_i\}$ zavedeme značení

$CC_{ij} \in \mathbb{R}^+$,	Objemová kapacita j-tého vozidla v i-tém vozovém parku.
$CW_{ij} \in \mathbb{R}^+$,	Váhová kapacita j-tého vozidla v i-tém vozovém parku.
$\Omega \in \mathbb{R}^+$.	Maximální doba provozu každého vozidla.

Dále definujme $CE_{ij} \in \mathbb{R}^+$jako dodatečné vybavení vozidla (jeřáb aj.). Tuto podmínku lze chápat tak, že konkrétnímu typu vybavení vozidla přiřadíme unikátní konstantu z $\mathbb{R}^+$. Objednávky, které vyžadují pro obsluhu vozidlo určitého typu, jsou charakterizovány podobnou konstantou, kterou uvedeme dále ($OE_i \in \mathbb{R}_0^+$).

Také zavedeme $CTI_{ij} \in \mathbb{R}^+$ (resp. $CCI_{ij} \in \mathbb{R}^+$), $i \in \{1,2,..,U\}$, $j \in \{1,2,..,K_i\}$, jako náklady na hodinu provozu (resp. náklady na kilometr provozu). Lze to chápat tak, že každé vozidlo má odlišnou spotřebu pohonných hmot a také se liší v průměrných časech přesunů mezi jednotlivými destinacemi. Konstanty CTI_{ij} a CCI_{ij} nám pomohou zohlednit spotřebu a časové přesuny daného vozidla.

V neposlední řadě zavedeme fixní časovou dotaci pro obsloužení každé objednávky jako $\varsigma \in \mathbb{R}^+$, (což je čas potřebný k obsloužení objednávky) a dále fixní časovou dotaci pro nakládku zboží do vozidla ve skladu zboží jako $\tau \in \mathbb{R}^+$ (což je čas potřebný k naložení zboží

ve skladu zboží). V praxi tyto časy závisí na množství zboží, nicméně my je pro jednoduchost uvažujeme jako fixní.

Každé vozidlo, které obsluhuje určitý počet objednávek, musí nejprve naložit distribuované zboží ve skladu zboží a až poté započne rozvážet. Jelikož vzdálenost vozových parků a uvažovaného skladu zboží je obecně různá, definujme časovou a distanční vzdálenost mezi skladem a i-tým vozovým parkem. Distanční vzdáleností chápeme vzdálenost mezi daným vozovým parkem a skladem, časovou vzdáleností dobu přesunu vozidla mezi daným vozovým parkem a skladem.

$ST_i \in \mathbb{R}_0^+$, Časová vzdálenost skladu a i-tého vozového parku.

$SC_i \in \mathbb{R}_0^+$. Distanční vzdálenost skladu a i-tého vozového parku.

Pro každou objednávku dále mějme dánu váhovou a objemovou kapacitu a dodatečná omezení pro výbavu vozidla. Uvažujme $i \in \{1,2, \dots, M\}$

$M \in \mathbb{N}$, Celkový počet objednávek.

..

$OC_i \in \mathbb{R}^+$, Objemová kapacita i-té objednávky.

$OW_i \in \mathbb{R}^+$. Váhová kapacita i-té objednávky.

Dále definujme $OE_i \in \mathbb{R}_0^+$, což jsou nutné podmínky na vybavení vozidla pro i-tou objednávku. Tuto konstantu definujeme stejně jako CE_{ij}, která byla zmíněna výše.

Pro následující definice mějme $i \in \{0,1,2, \dots, M\}$, $j \in \{0,1,2, \dots, M\}$ a $i \neq j$. Index $i = 0$ nebo $j = 0$ označuje sklad zboží

$DT_{ij} \in \mathbb{R}_0^+$, Časová vzdálenost mezi i-tou a j-tou objednávkou (příp. skladem).

$DC_{ij} \in \mathbb{R}_0^+$. Distanční vzdálenost mezi i-tou a j-tou objednávkou (příp. skladem).

Předpokládejme, že množina objednávek je konečná a časové i distanční vzdálenosti mezi jednotlivými objednávkami (příp. i mezi skladem a objednávkami a mezi objednávkami

a vozovými parky) jsou známé. Nyní definujme pomocné proměnné. Předpokládejme, že jejich hodnoty jsou nastaveny tak, aby řešení modelu bylo přípustné, to znamená, aby jednotlivé rozvozní trajektorie vozidel byly kruhové, vozidlo navštívilo jako první destinaci své trasy sklad zboží a jako poslední destinaci výchozí vozový park. Je dovoleno, aby vozidlo jelo více okruhů, v tom případě se do místa skladu zboží vrací vícekrát a vždy zde začíná nový okruh. Poslední okruh je zakončen ve výchozím vozovém parku.

Pro následující definice mějme $i \in \{1,2,\dots,U\}$, $j \in \{1,2,\dots,|K_i|\}$, $k \in \{1,2,\dots,M\}$, $m \in \{0,1,2,\dots,M\}$ a $n \in \{0,1,2,\dots,M\}$. Pokud zvolíme m-tý nebo n-tý index nulový, pak jde o sklad zboží.

$X_{ijk} \in \{0,1\}$, ... Pomocná proměnná, která vyjadřuje, zda j-té vozidlo z i-tého vozového parku obsluhuje k-tou objednávku.

$Y_{ij} \in \{0,1\}$, ... Pomocná proměnná, která vyjadřuje, zda j-té vozidlo z i-tého vozového parku bude při rozvozním problému využito.

$U_{ijmn} \in \{0,1\}$, ... Pomocná proměnná, která vyjadřuje, zda j-té vozidlo i-tého vozového parku projede trasu z m-té objednávky do n-té objednávky.

$V_{ij} \in \{0,1,2,\dots,M\}$. ... Pomocná proměnná, která vyjadřuje, kolik okružních jízd ze skladu zboží pojede j-té vozidlo i-tého vozového parku.

Pro každou objednávku musí platit, že je obsloužena právě jedním vozidlem. Taktéž poslední objednávka pro každé vozidlo je právě jedna:

$$\sum_{k=1}^{M} X_{ijk} = 1, \qquad \text{(R 3.3)}$$

kde $i \in \{1,2,\dots,U\}$, $j \in \{1,2,\dots,|K_i|\}$. Zde je nutno připomenout důležitý předpoklad, že pro každou destinaci existuje vozidlo s takovou kapacitou, aby byla destinace obsloužena (tudíž neuvažujeme využití více vozidel pro obsloužení jedné destinace). Objednávku, kterou nelze obsloužit jedním vozidlem rozdělíme na více objednávek tak, aby jednotlivé dílčí

objednávky splňovaly uvedený předpoklad. Tento předpoklad není na újmu obecnosti a uvažovaný model je tedy korektní.

V neposlední řadě je třeba ověřit kapacitní podmínky jednotlivých vozidel.

Pro každé vozidlo, které rozváží zboží pro dané objednávky, musí platit, že nejsou překročeny jeho kapacitní možnosti, aneb že suma kapacit všech objednávek přiřazených tomuto vozidlu nepřekračuje kapacitu tohoto vozidla. Pro toto formální vyjádření je použit pouze m-násobek kapacity vozidla, kde $m \in \mathbb{N}$ je počet okruhů:

$$m \cdot Y_{ij} \cdot CW_{ij} \geq \sum_{k=1}^{M} \left(X_{ijk} \cdot OW_k\right), \tag{R 3.4}$$

$$m \cdot Y_{ij} \cdot CC_{ij} \geq \sum_{k=1}^{M} \left(X_{ijk} \cdot OC_k\right). \tag{R 3.5}$$

Kde $i \in \{1,2,..,U\}$, $j \in \{1,2,..,K_i\}$.

Tyto podmínky lze chápat tak, že vozidlo je schopno pokrýt svými kapacitními možnostmi všechny objednávky v maximálně $m \in \mathbb{N}$ okruzích.

Pokud některé objednávky požadují speciální podmínky na vybavení vozidla, pak musí vozidlo, které je obsluhuje, tyto podmínky splňovat. Pro každou objednávku jsou splněna dodatečná omezení na vozidlo, které ji obsluhuje:

$$X_{ijk} \cdot CE_{ij} \geq X_{ijk} \cdot OE_k, \tag{R 3.6}$$

Kde $i \in \{1,2,\dots,U\}$, $j \in \{1,2,\dots,|K_i|\}$, $k \in \{1,2,\dots,M\}$.

Pro každé vozidlo musí platit, že jeho provozní doba nepřekročí maximální dobu provozu vozidla $\Omega \in \mathbb{R}^+$

$$2 \cdot ST_i + V_{ij}\tau + \sum_{m=1}^{M} \sum_{\substack{n=0,\\ m \neq n}}^{M} U_{ijmn}(\varsigma + DT_{mn}) + \sum_{m=1}^{M} U_{ij0m} DT_{0m} \leq \Omega, \tag{R 3.7}$$
$$i \in \{1,2,\dots,U\}, j \in \{1,2,\dots,|K_i|\}.$$

Účelová funkce je potom určena jako minimalizace časových nákladů:

$$\sum_{i=1}^{U}\sum_{j=1}^{|K_i|} Y_{ij}\left(2\cdot ST_i + V_{ij}\tau + \sum_{m=1}^{M}\sum_{\substack{n=0,\\ m\neq n}}^{M} U_{ijmn}(\varsigma + DT_{mn}) + \sum_{m=1}^{M} U_{ij0m}DT_{0m}\right)\overset{min}{\longrightarrow}. \quad \text{(R 3.8)}$$

Při formulaci účelové funkce postupujeme následovně: Pro každé vozidlo z každého vozového parku vyčíslíme časové náklady na přesun vozidla do skladu, poté přičteme časy všech nakládek zboží pro každý okruh a konečně časové náklady jednotlivých okruhů.

Někdy je požadována minimalizace finančních nákladů:

$$\sum_{i=1}^{U}\sum_{j=1}^{|K_i|} Y_{ij}\left(2\cdot SC_i + \sum_{m=0}^{M}\sum_{\substack{n=0,\\ m\neq n}}^{M} U_{ijmn}(DC_{mn})\right)\overset{min}{\longrightarrow}. \quad \text{(R 3.9)}$$

3.2 Postup řešení rozšířeného dopravního problému

Jelikož v rozšířeném modelu dopravního problému (viz paragraf 3.1) uvažujeme konečný počet základen, které mohou ve svých vozových parcích vlastnit obecně různý konečný počet vozidel. Je třeba rozdělit objednávky k jednotlivým základnám, potažmo k jejich vozidlům. Z hlediska optimalizace nákladů na dopravu je žádoucí, aby počet využitých vozidel byl minimální, ale zároveň aby každé vozidlo kapacitně pokrývalo jemu přidělené objednávky. Metodiku lze rozdělit do tří fází.

V první fázi se zaměříme na rozdělení objednávek pro jednotlivé základny. Takto získáme disjunktní množiny objednávek pro jednotlivé základny, pro něž musí platit podmínka úplnosti

$$\begin{gathered} B_i \cap B_j = \emptyset\,, \\ i,j \in \{1,2,\dots,O\}, i \neq j, \end{gathered} \quad \text{(R 3.10)}$$

$$BC_i \leq \Phi_{\mathrm{i}}, \quad \text{(R 3.11)}$$

$$\bigcup_{i=1}^{O} B_i = Q, \quad \text{(R 3.12)}$$

kde B_i je množina objednávek přiřazená i-té základně, $O \in \mathbb{N}$ je počet základen, BC_i je součet objemových kapacit všech objednávek, které jsou přiřazeny i-té základně, $\Phi_i \in \mathbb{R}^+$ je objemová kapacita i-té základny, Q je množina všech objednávek. Tyto podmínky lze chápat tak, že každá objednávka je přiřazena právě jedné základně a součet objemových kapacit jednotlivých objednávek přiřazených dané základně nesmí překročit objemovou kapacitu dané základny.

V druhé fázi metodiky se zaměříme na rozdělení objednávek ke konkrétním vozidlům, pro něž můžeme formulovat obdobné podmínky jako pro první fázi:

$$C_k \cap C_l = \emptyset, \quad k, l \in \{1,2,\dots,N\}, k \neq l, \tag{R 3.13}$$

$$\bigcup_{k=1}^{N} C_k = W. \tag{R 3.14}$$

Zde C_k zastupuje množinu objednávek přiřazenou k-tému vozidlu v určité základně, N je počet vozidel a konečně W je množina objednávek pro tuto základnu. Zřejmě tedy platí $\big(\exists(i \in \{1,2,\dots,O\}): (B_i \equiv W)\big)$.

Konečně v poslední fázi je třeba určit trasy jednotlivých vozidel tak, aby každé vozidlo obsloužilo jemu přiřazené objednávky a aby distribuční náklady na přepravu zboží byly co nejnižší.

3.3 Návrhy řešení vybraných dopravních problémů

V následujících paragrafech popíšeme navržené metody pro řešení jednotlivých fází, které byly popsány v paragrafu 3.2. Navržené metody rozdělíme dle vybraných dopravních problémů a dále v kapitole 4 je otestujeme na reálných datech.

3.3.1 *Rozdělení objednávek pro jednotlivé základny*

Při řešení reálných dopravních problémů se touto fází obvykle není nutné zabývat, neboť objednávky jsou k základnám obvykle přiřazeny na základě okresní (nebo krajské) příslušnosti k základně, případně na základě předchozích zkušeností.

Pro rozdělení objednávek pro jednotlivé základny lze využít modifikaci Jarníkova algoritmu pro hledání minimální kostry ohodnoceného grafu [15]. Dle značení v paragrafu 3.2 zavedeme $\Phi_i \in \mathbb{R}^+$ jako objemovou kapacitu i-té základny, $i \in \{1,2,\dots,U\}$. Za počáteční množinu objednávek přiřazenou určité základně volíme všechny objednávky, jejichž vzdálenost od této základny je nižší nebo rovna definované konstantě $\lambda \in \mathbb{R}^+$. Tuto určíme v intervalu $\langle 0; \frac{D^*}{2}\rangle$, kde $D^* \in \mathbb{R}^+$ je vzdálenost uvažované základny k nejbližší další základně. Zároveň musí být splněno, že součet objemových kapacit všech objednávek pro každou základnu je nižší nebo roven její objemové kapacitě (dále jen podmínka objemové úplnosti). Je patrné, že modifikací konstanty λ lze měnit chování celého algoritmu, neboť upravením této konstanty může dojít k přeuspořádání počátečních množin objednávek přiřazených k jednotlivým základnám.

Nejprve jsou objednávky, jejichž vzdálenost od nejbližší základny je nižší než stanovená konstanta λ pro tuto základnu, přidány do množiny objednávek přidělených pro tuto základnu. Pro řešení problému v Eukleidovském prostoru je zřejmé, že při tomto postupu je každá objednávka přidána nejvýše do jedné množiny přiřazené určité základně.

Objednávky, které zůstanou nepřiřazeny, jsou procházeny v náhodném pořadí a pro každou nepřiřazenou objednávku $\mathcal{O}$ určíme nejbližší objednávku $\mathcal{P}$, která je přiřazena určité základně. Pokud nalezneme více přiřazených objednávek se stejnou vzdáleností k $\mathcal{O}$, pak prioritně vybíráme takovou objednávku $\mathcal{P}$, která je nejblíže k základně jí přiřazené. Je-li objednávka $\mathcal{P}$ obsažena v k-té množině, pak přidáme objednávku $\mathcal{O}$ taktéž do k-té množiny a iterujeme postup s další nepřiřazenou objednávkou. Stále je zapotřebí uvažovat podmínku objemové úplnosti pro danou základnu. Podobný postup lze vysledovat právě v popisu Jarníkova algoritmu pro hledání minimální kostry ohodnoceného grafu.

Korektnost celého postupu lze snadno dokázat matematickou indukcí. V první fázi rozdělíme objednávky do disjunktních množin přiřazeným k základnám tak, že každá

základna má přiřazeny objednávky, jejichž vzdálenost od dané základny je nižší než stanovená konstanta λ. Následně v každé iteraci algoritmu náhodně vybereme nepřiřazenou objednávku $\mathcal{O}$ a přiřadíme ji do té množiny, která obsahuje vzdálenostně nejbližší přiřazenou objednávku k $\mathcal{O}$. V každé iteraci se tedy sníží počet nepřiřazených objednávek o 1. Proto s počátečním předpokladem disjunktních množin uvedený postup splňuje podmínky úplnosti definované výše v úvodu paragrafu 3.2.

3.3.2 *Rozdělení objednávek pro jednotlivá vozidla*

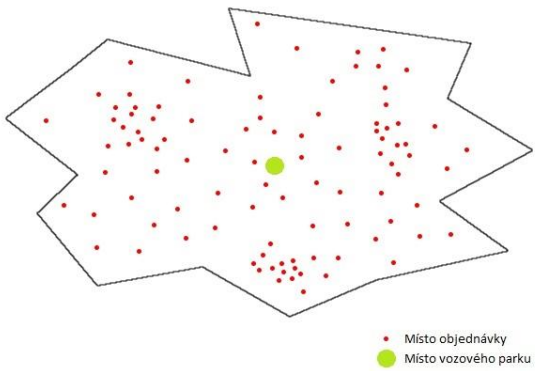
V druhé části algoritmu se zabýváme rozdělením objednávek pro jednotlivá vozidla ve sledované základně tak, abychom využili co nejméně vozidel s co nejnižšími provozními náklady. Zde lze rozdělit případy dle umístění vozového parku vůči objednávkám.

Pro jednoduchost předpokládáme, že objednávky jsou četnější v blízkosti větších měst, (jinak uvažujeme normální rozdělení). Tento předpoklad sice není vždy reálný, nicméně nelze obecně popsat variabilitu rozdělení objednávek a proto tuto aproximaci považujeme za dostatečnou.

V následujících paragrafech uvedeme nejčastější případy umístění vozového parku, které v praxi mohou nastat.

3.3.2.1 *Středová varianta*

Ve středové variantě je vozový park rovnoměrně obklopen objednávkami (viz Obr. 3.1).

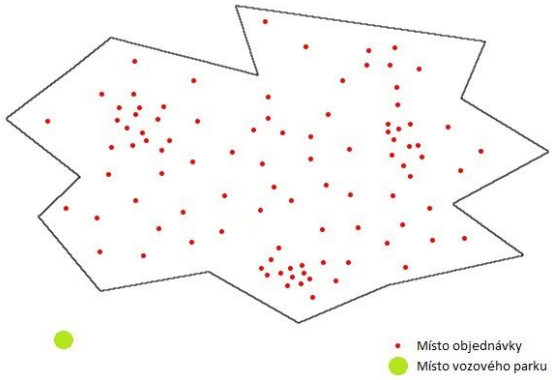


Obr. 3.1 Středová varianta umístění vozového parku

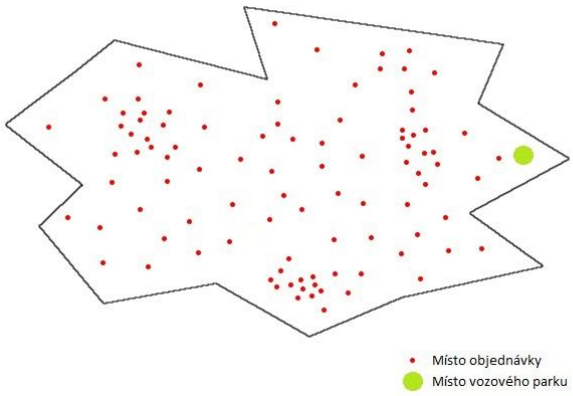
Pro tuto variantu se zdá být výhodné přiřazovat objednávky k vozidlům na základně kapacity.

3.3.2.2 *Krajní varianta*

V krajní variantě je vozový park situován na okraji (případně vně) území, na kterém jsou rovnoměrně rozmístěny objednávky (viz Obr. 3.2 nebo Obr. 3.3).



Obr. 3.2 Krajní varianta umístění vozového parku (vně)

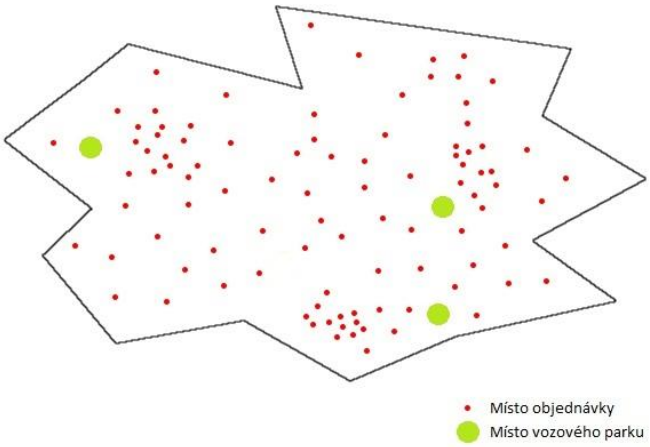


Obr. 3.3 Krajní varianta umístění vozového parku (hraniční)

Tato varianta musí počítat s větší dojezdovou vzdáleností k objednávkám. Z tohoto důvodu se zdá být výhodné přiřazovat objednávky nejen na základě kapacity, ale i dojezdové vzdálenosti.

3.3.2.3 *Varianta více vozových parků*

Tato varianta uvažuje větší počet vozových parků, které mohou být umístěny jak středově (středová varianta) vzhledem k objednávkám, tak krajně (krajní varianta).



Obr. 3.4 Varianta více vozových parků

Jednotlivé objednávky přiřadíme vozovým parkům dle algoritmu rozdělení objednávek základnám popsaného v paragrafu 3.3.1. Pro každý vozový park poté postupujeme dle středové, příp. krajní varianty.

3.3.2.4 *Návrh metody řešení pro objednávky bez speciálních podmínek*

V tomto případě uvažujeme případ přiřazení vozidel k objednávkám, které nevyžadují speciální podmínky pro obsluhující vozidlo. Jedná se především o zjednodušené případy rozvozu zboží (rozvoz piva, volebních lístků, plynu aj.). Objednávky nemají speciální omezení na vybavení, příp. kapacitu vozidla.

Nejprve je třeba provést odhad počtu vozidel, která budou využita pro obsloužení objednávek přiřazených k danému vozovému parku.

Samotný odhad lze provést např. prostým pokrytím všech objednávek. Seřadíme vozidla dle uživatelských preferencí (příp. nákladů na provoz) a postupujeme tak, že postupně vybíráme vozidla v pořadí dle seznamu a pro každé vozidlo sestavíme náhodně maximální podmnožinu objednávek takovou, aby ji vozidlo kapacitně pokrylo (chápejme ve smyslu splnění všech restrikcí pro dané vozidlo a objednávky). Takto postupujeme, dokud nejsou

pokryty všechny objednávky nebo dokud není vozový park plně využit. Konečný počet využitých vozidel prohlásíme jako korektní odhad.

Odhad počtu vozidel lze také provést některou z metod řešení přiřazovacího problému, jež byl popsán v paragrafu 2.3. V případě různé kapacity vozidel lze použít Mayerovu metodu popsanou v paragrafu 2.3.1. Obecně se jedná o NP-úplnou úlohu již pro případ shodné kapacity všech vozidel vozového parku.

3.3.2.4.1 Středová varianta rozložení vozového parku

Předpokládejme nyní, že vozový park je umístěn podobně jako ve středové variantě popsané v paragrafu 3.3.2.1 a že kapacity vozidel jsou přibližně stejné.

Seřadíme objednávky přiřazené k danému vozovému parku podle jejich velikosti. Předpokládejme, že $\omega \in \mathbb{N}$ je odhad počtu vozidel. Vybereme prvních ω objednávek a přiřadíme je postupně k ω vozidlům z vozového parku, která řadíme dle uživatelských preferencí (např. náklady na provoz). Opakujeme postup pro dalších ω objednávek dokud nejsou vyčerpány všechny objednávky, přičemž vozidla kapacitně pokrývají jim přiřazené objednávky. Pokud vozidlo nemůže kapacitně pokrýt objednávku, je objednávka pokryta dalším vozidlem v pořadí.

V případě, že jsou kapacitní možnosti vozidel různé, použijeme metodu popsanou pro krajní variantu umístění vozového parku popsanou v paragrafu 3.3.2.4.2.

Korektnost postupu z hlediska podmínek úplnosti (definovaných v úvodu paragrafu 3.2) lze snadno nahlédnout. V každé iteraci algoritmu je přiřazena alespoň jedna nepřiřazená objednávka, přičemž pokud nastane situace, kdy objednávku nelze kapacitně pokrýt žádným z vozidel uvažovaných při odhadu, přidáme další vozidlo z vozového parku (pokud existuje). Maximálně po M ($M \in \mathbb{N}$ je celkový počet objednávek) iteracích jsou vozidlům přiřazeny všechny objednávky příslušné k tomuto vozovému parku.

3.3.2.4.2 Krajní varianta rozložení vozového parku

Pokud je vozový park umístěn podobně jako v krajní variantě popsané v paragrafu 3.3.2.2, pak lze pro řešení tohoto problému využít Mayerovu metodu popsanou v paragrafu 2.3.1. Ta s využitím Jarníkova algoritmu pro hledání minimální kostry ohodnoceného grafu [15] řeší problém z hlediska optimalizace kapacitního obsazení vozidel.

Seřadíme vozidla dle uživatelských preferencí (např. nákladů na provoz) a vybereme první vozidlo s nejvyšší preferencí a nejvzdálenější objednávku od sledované základny, která dosud není přiřazená žádnému vozidlu. Tuto přidáme do množiny přiřazené vybranému vozidlu. Dále iterativně volíme další objednávku takovou, která není přiřazena žádnému vozidlu a zároveň má minimální vzdálenost od množiny objednávek přiřazených vybranému vozidlu. Tuto objednávku přidáme do této množiny. Pokud vozidlo nedokáže kapacitně pokrýt další takto vybranou objednávku, vybereme další vozidlo v pořadí a opakujeme postup přiřazování objednávek tak dlouho, dokud existuje alespoň jedna objednávka nepřiřazena žádnému vozidlu (vybrané vozidlo vždy kapacitně pokrývá přiřazené objednávky).

Je výhodné, že uvedený postup nepotřebuje podmínku stejné kapacity všech vozidel, tudíž lze plně využít transportní možnosti každého vozidla.

Korektnost postupu z hlediska podmínek úplnosti definovaných v úvodu paragrafu 3.2 lze snadno nahlédnout. V každé iteraci algoritmu je přiřazena alespoň jedna objednávka, přičemž pokud nastane situace, kdy objednávku nelze kapacitně pokrýt žádným z vozidel uvažovaných při odhadu, přidáme další vozidlo z vozového parku (pokud existuje). Maximálně po M ($M \in \mathbb{N}$ je celkový počet objednávek) iteracích jsou přiřazeny všechny objednávky příslušné k tomuto vozovému parku.

3.3.2.4.3 Varianta více vozových parků

Dle algoritmu rozdělení objednávek k jednotlivým základnám popsaného v paragrafu 3.3.1 rozdělíme objednávky k vozovým parkům. Pro každý vozový park poté postupujeme dle středové (resp. krajní) varianty popsané v paragrafu 3.3.2.1 (resp. 3.3.2.2).

3.3.2.5 *Návrh metody řešení pro objednávky se speciálními podmínkami*

V následujícím paragrafu budeme uvažovat objednávky se speciálními podmínkami, které byly definovány v modelu rozšířeného dopravního problému v rámci paragrafu 3.1. Přiřazujeme-li vozidla k objednávkám se speciálními podmínkami, musí každé vozidlo splnit podmínky pro objednávky jemu přiřazené.

V následujícím popisu algoritmu se budeme držet značení definovaného v paragrafu 3.1. Následující metodu lze použít jak pro krajní, tak i pro středové rozložení vozového parku k jednotlivým objednávkám.

Nejprve vybereme vozidlo s nejnižšími náklady na provoz a budeme mu postupně přiřazovat objednávky a vytvářet tak okruhy pro toto vozidlo. Jakmile časové odhady okruhů překročí maximální dobu provozu vozidla, přidáme další vozidlo a opakujeme postup přiřazení objednávek a konstrukce okruhů. Algoritmus končí, když jsou přiřazeny všechny objednávky, případně pokud neexistuje další volné vozidlo.

Předpokládejme, že vozidla jsou řazena prioritně dle nákladů na kilometr provozu vzestupně, dále pak dle nákladů na hodinu provozu sestupně. Dále předpokládejme, že objednávky jsou řazeny dle speciálních podmínek $OE \in \mathbb{R}_0^+$ sestupně.

Vybereme první vozidlo C_i ze seznamu vozidel řazeného dle výše uvedených priorit. Vozidlu C_i přiřadíme první objednávku O_j ze seznamu objednávek (řazeného dle výše uvedených priorit), pro kterou vozidlo C_i splňuje speciální podmínky ($CE_i \geq OE_j$) a zároveň kapacita vozidla C_i je vyšší nebo rovna kapacitě objednávky O_j ($CC_i \geq OC_j, CW_i \geq OW_j$), kde $i \in \{1,2,\dots,N\}$, $j \in \{1,2,\dots,M\}$, $N \in \mathbb{N}$ je počet vozidel a $M \in \mathbb{N}$ je počet objednávek.

Dále hledáme vzdálenostně nejbližší objednávku O_m k množině objednávek přiřazených vozidlu C_i, jejíž kapacita nepřekročí kapacitu vozidla a zároveň takovou, že dané vozidlo splňuje speciální podmínky pro tuto objednávku ($CE_i \geq OE_m$), $m \in \{1,2,\dots,M\}$.

Pokud existuje, přiřadíme tuto objednávku vozidlu C_i a iterativně opakujeme postup popsaný v předchozím odstavci.

Vypočteme časový odhad obsloužení objednávek přiřazených pro tento okruh vozidla a (pokud suma časových odhadů všech okruhů daného vozidla nepřekračuje maximální provozní čas vozidla) odebereme objednávky přiřazené v tomto okruhu ze seznamu objednávek a iterujeme celý postup algoritmu pro vozidlo C_i. Jinak odebereme vozidlo C_i ze seznamu vozidel a iterujeme celý postup algoritmu.

Metoda vytvoří jednotlivé okruhy pro vozidla, přičemž preferuje vozidla s nízkými náklady na provoz a prioritně jim přiřazuje objednávky se speciálními podmínkami.

3.3.3 *Konstrukce trasy pro jednotlivá vozidla*

V následujícím paragrafu shrneme navrhované metody pro konstrukci tras pro jednotlivá vozidla tak, aby byly obslouženy všechny jim přiřazené objednávky. Metody rozdělíme dle počtu objednávek přiřazených k danému vozidlu. Pro „malé“ rozsahy lze určit optimální řešení tohoto problému, zatímco s větším počtem objednávek narůstá výpočetní složitost a je nutné použít některou z heuristických metod.

Dále popíšeme různé účelové funkce, jejichž minimum budeme hledat.

3.3.3.1 *Minimalizační funkce*

Pro samotný průběh algoritmu je nutné zvolit účelovou funkci (jejíž minimum budeme hledat), takovou, aby byla vhodná pro dané rozložení objednávek.

Zde uvádíme několik variant minimalizačních funkcí:

a) Minimalizace i vzhledem k návratu vozidel do vozového parku. Při této volbě účelové funkce minimalizujeme časové náklady pro obsloužení všech objednávek a také přihlížíme k návratu vozidla do výchozího vozového parku.
b) Minimalizace bez návratu do vozového parku. Tato účelová funkce nebere v úvahu návrat vozidla do vozového parku a minimalizuje pouze časové náklady pro obsloužení všech objednávek.
c) Parametrická minimalizace. Předpokládá uživatelsky definovaný parametr $\lambda \in (0; \infty)$. Tato varianta předpokládá přidávání objednávek do stávajícího

okruhu dle vzdálenosti těchto objednávek od okruhu. Pokud je tato vzdálenost menší nebo rovna parametru λ, pak se použije varianta minimalizace bez návratu do vozového parku (viz b)), v opačném případě varianta minimalizace i vzhledem k návratu vozidel do vozového parku (a). Takto lze parametricky měnit chování účelové funkce dle vzdálenosti uvažovaných objednávek.

Mohlo by se zdát výhodné volit vždy variantu minimalizace s návratem vozidla do vozového parku, ale musíme si uvědomit, že uvedený algoritmus provádí výpočet pro každé vozidlo zvlášť a tudíž ve výsledku můžeme dostat vyšší celkovou hodnotu účelové funkce než pro jinou variantu.

Pokud předpokládáme okrskově situované rozložení (tzn. takové, kdy objednávky jsou četnější v obydlených oblastech, jinde pouze výjimečně), pak se zdá výhodné volit parametrickou minimalizaci s nízkou hodnotou parametru λ, příp. minimalizaci bez návratu do vozového parku. Tato úvaha vychází z předpokladu, že pokud vozidlo obsluhuje oblast s větší četností objednávek, pak jako další vybírá vždy objednávky z této oblasti (dokud vozidlo tuto množinu objednávek kapacitně nepokryje).

Naopak pokud uvažujeme rovnoměrné rozdělení (tj. takové rozdělení, kdy objednávky jsou náhodně rozmístěny po distribuční mapě), pak může být výhodné volit parametrickou minimalizaci s vysokou hodnotou parametru λ, příp. minimalizaci s návratem do vozového parku.

Další možností zohlednění rozložení objednávek je střídání vozidel v přidělování jednotlivých objednávek. Dosud jsme uvažovali postup, kdy vybranému vozidlu přiřazujeme objednávky, dokud jsou splněna jeho kapacitní omezení. V případě, že vozidlo již není schopno kapacitně obsloužit další objednávku, volíme další vozidlo a opakujeme postup přiřazování objednávek. Objednávky lze ale přiřazovat i tak, že pro každou objednávku určíme vhodné vozidlo. Tato varianta se zdá technicky náročnější, nicméně pokud lze volit různá vozidla pro každou objednávku, je možné v určitých případech docílit lepšího rozdělení množin objednávek k jednotlivým vozidlům (potažmo také k nižší hodnotě výsledné účelové funkce).

Střídání vozidel v přidělování objednávek je asi výhodnější v případě rovnoměrného rozdělení objednávek na distribuční mapě, nicméně tento předpoklad je nutno důkladně otestovat.

Mezi nevýhody této varianty bychom zařadili fakt, že metoda neověřuje počáteční odhad počtu vozidel a tudíž nejsme schopni zpětně ověřit, zda byl výhodný.

3.3.3.2 *Postup pro malé rozsahy*

Tento postup využívá metodu dynamického programování. Trasu vozidla zkonstruujeme dle postupu, který jsme popsali v paragrafu 2.1.1.1. Navíc zde volíme minimalizační funkci dle preferencí z paragrafu 3.3.3.1. Takto lze algoritmus přizpůsobit uvažovanému rozložení objednávek k vozovému parku.

Metoda dynamického programování se zdá být efektivní pro problémy, kdy uvažujeme maximálně 20 objednávek. Pro větší počet je vhodné použít postup pro velké rozsahy popsaný dále.

3.3.3.3 *Postup pro velké rozsahy*

Pokud je počet objednávek větší než 20, projevuje se negativně výpočetní složitost metody dynamického programování. Proto tento postup nahrazujeme modifikací Jarníkova algoritmu [15] upraveného dle potřeb řešení úlohy obchodního cestujícího.

Nejprve dle uvažovaného rozložení objednávek vzhledem k vozovému parku zvolíme minimalizační funkci dle popisu v paragrafu 3.3.3.1.

Metoda konstruuje trajektorii vozidla. Definujme množinu objednávek $\mathcal{A}$ obsažených v trajektorii vozidla. Na začátku je $\mathcal{A} = \emptyset$.

V případě minimalizační funkce vzhledem k návratu vozidla do vozového parku hledáme takovou objednávku, pro niž součet vzdálenosti této objednávky od množiny $\mathcal{A}$ a vzdálenosti této objednávky od vozového parku je minimální. Naopak pokud volíme minimalizační funkci bez návratu vozidla do vozového parku, hledáme objednávku s minimální vzdáleností od množiny $\mathcal{A}$. Uvažovanou objednávku přidáme do trajektorie vozidla.

Iterativně opakujeme přiřazení objednávek výše popsaným způsobem, dokud nejsou v trajektorii vozidla obsaženy všechny objednávky přiřazené tomuto vozidlu.

3.3.4 *Složitost*

Složitost celého algoritmu je velice závislá na použitých implementačních strukturách, přesto zde uvedeme velice jednoduché odhady složitosti jednotlivých fází algoritmu.

Při rozdělení objednávek k jednotlivým základnám postupujeme iteračně přes všechny objednávky. Pro každou objednávku je nutné určit, k jaké základně bude přiřazena a proto složitost této fáze metody odhadneme jako $O(M \cdot B)$, kde $M \in \mathbb{N}$ je počet objednávek a $B \in \mathbb{N}$ je počet základen.

Dále uvažujeme rozdělení objednávek (bez speciálních podmínek) k jednotlivým vozidlům. Algoritmus popsaný v paragrafu 3.3.2.4.1 pracuje s lineární složitostí až na řazení objednávek a vozidel na začátku algoritmu. Tudíž složitost lze odhadnout jako $O(M \cdot logM + N \cdot logN)$, kde $N \in \mathbb{N}$ je počet vozidel. Algoritmus popsaný v paragrafu 3.3.2.4.2 je modifikací Jarníkova algoritmu a proto ho odhadneme složitostí $O(M^2)$ stejně jako Jarníkův algoritmus.

Uvažujeme-li rozdělení objednávek (se speciálními podmínkami) k jednotlivým vozidlům, navržený algoritmus popsaný v paragrafu 3.3.2.5 lze také odhadnout složitostí $O(M^2)$, neboť se jedná o další speciální modifikaci Jarníkova algoritmu.

Konstrukce trasy pro malé rozsahy objednávek využívá metodu dynamického programování, proto pro všechny vozidla odhadneme složitost jako $O(N \cdot M^2 \cdot 2^M)$. Konstrukce trasy pro velké rozsahy využívá opět modifikaci Jarníkova algoritmu, a tudíž pro všechny vozidla lze stanovit odhad jako $O(N \cdot M^2)$.

4 Implementace

V následujících paragrafech aplikujeme algoritmy navržené v rámci paragrafu 3 na vybrané reálné dopravní problémy. Jedná se konkrétně o problém rozvozu lístků pro sčítání lidu z Prahy do krajských měst České republiky. Dále problém maloobchodní distribuce potravin z centrálního skladu v Prostějově do většího množství míst v rámci České republiky. Závěr kapitoly věnujeme srovnání s ostatními algoritmy, které se již používají v praxi.

4.1 Problém rozvozu lístků sčítání lidu

Následující problém je převzat z publikace [35].

Jedná se o problém rozvozu lístků sčítání lidu z Prahy do všech krajských měst České republiky. Jedná se konkrétně o města Brno (Br), České budějovice (ČB), Hradec Králové (HK), Jihlava (Ji), Karlovy Vary (KV), Liberec (Li), Olomouc (Ol), Ostrava (Os), Pardubice (Pa), Plzeň (Pl), Ústí nad Labem (ÚnL) a Zlín (Zl).

4.1.1 *Data*

Matice vzdáleností a jednotlivé kapacity krajských měst byly převzaty z publikace [35]. Data jsou přehledně shrnuta v následující tabulce

Místo	Kapacity	Vzdálenosti											
		Os	Zl	Ol	Br	ČB	KV	Ji	HK	Pa	Li	Pl	ÚnL
Pr	-	362	297	285	202	140	133	123	112	104	102	94	92
Os	1278	-	104	93	165	346	495	253	240	240	335	456	454
Zl	598		-	63	100	281	430	188	212	210	309	391	389
Ol	641			-	78	259	408	166	149	147	246	369	367
Br	1136				-	186	335	93	142	138	239	296	294
ČB	626					-	216	126	217	196	242	133	232
KV	304						-	256	245	237	205	83	122
Ji	521							-	110	89	185	186	215
HK	551								-	21	97	206	166
Pa	509									-	118	198	196
Li	429										-	196	92
Pl	551											-	146
ÚnL	827												-

Tab. 4.1 Kapacity a vzdálenosti mezi krajskými městy pro problém rozvozu lístků pro sčítání lidu

kde kapacitou krajského města chápeme počet požadovaných lístků sčítání lidu v tisících a vzdálenosti mezi jednotlivými krajskými městy udáváme v kilometrech.

Předpokládejme, že máme k dispozici jediné vozidlo kapacity 2 100 000 lístků sčítání lidu. Vozidlo absolvuje každý svůj okruh z centrálního stanoviště Praha a končí také v Praze.

Cílem problému je obsloužení všech krajských měst, přičemž požadujeme minimalizaci ujetých kilometrů.

4.1.2 *Rozložení objednávek*

V následujícím paragrafu se zaměříme na analýzu rozložení objednávek (krajských měst) vzhledem k umístění centrálního stanoviště Praha. Samotnou konstrukci provedeme pomocí mapových podkladů Google Earth.

Obr. 4.1 Rozložení krajských měst vzhledem k Praze pro problém rozvozu lístků pro sčítání lidu

Z Obr. 4.1 je zřejmé, že pro rozložení objednávek vzhledem k depu nejvíce odpovídá středová varianta popsaná v paragrafu 3.3.2.1. V rámci testování použijeme i krajní variantu popsanou v paragrafu 3.3.2.2.

4.1.3 *Kontrukce okruhů*

Při konstrukci okruhů budeme postupovat dle metod navržených pro obě varianty rozložení vozového parku popsané v paragrafu 3.3.2.4.1 a 3.3.2.4.2 .

Nejprve provedeme odhad počtu okruhů.

Postupně přiřazujeme krajská města do okruhu dle výchozího řazení, dokud kapacity těchto měst nepřekročí kapacitu vozidla, tj. 2 100 000 lístků sčítání lidu.

Pokud jsou přiřazeny všechny objednávky, pak skončíme. Jinak přidáme další okruh a iteračně opakujeme postup přiřazování objednávek popsaný v odstavci zmíněném výše.

Pro tento případ jsme odhadli počet okruhů jako 5, přičemž první okruh obsahuje města Ostrava a Zlín, druhý Olomouc a Brno, třetí České Budějovice, Karlovy Vary, Jihlava a Hradec Králové, čvrtý Pardubice, Liberec a Plzeň a konečně poslední okruh obsahuje pouze město Ústí nad Labem.

4.1.3.1 *Středová varianta rozložení vozového parku*

Návrh metody pro konstrukci okruhů pro středovou variantu rozložení vozového parku je popsán v paragrafu 3.3.2.4.1.

Dle navržené metody bylo zkonstruováno 5 okruhů:

- Ostrava, Zlín
- Brno, Hradec Králové
- Ústí nad Labem, Plzeň, Liberec
- Olomouc, Jihlava, Karlovy Vary
- České Budějovice, Pardubice

Z popisu metody v paragrafu 3.3.2.4.1 je zřejmé, že středová varianta rozložení uvažuje pouze kapacity krajských měst a nikoli jejich vzdálenost. Metoda konstruuje okruhy, kdy jsou vozidla vytížena podobně, nicméně konstruované trajektorie pro tyto okruhy mohou být vzdálenostně náročné.

4.1.3.2 *Krajní varianta rozložení vozového parku*

Návrh metody pro konstrukci okruhů pro krajní variantu rozložení vozového parku je popsán v paragrafu 3.3.2.4.2.

Dle navržené metody bylo zkonstruováno 5 okruhů:

- Ostrava, Olomouc
- Zlín, Brno
- České Budějovice, Jihlava, Pardubice
- Karlovy Vary, Plzeň, Ústí nad Labem
- Hradec Králové, Liberec

Okruhy jsou konstruovány na základě vzdálenosti od centrálního stanoviště, tudíž se zdá, že vozidla v jednotlivých okruzích urazí menší vzdálenost. Na druhou stranu metoda neuvažuje podobné vytížení vozidel.

4.1.4 *Konstrukce trasy*

Z popisu konstrukce trasy pro jednotlivá vozidla (3.3.3) je zřejmé, že nejprve musíme zvolit metodu minimalizace (3.3.3.1) a vzhledem k počtu destinací zvolit buď metodu pro malé rozsahy (3.3.3.2), příp. metodu pro velké rozsahy (3.3.3.3). V průběhu testování bylo zjištěno, že postup pro malé rozsahy je neefektivní pro počty destinací větší než 11.

Jelikož počet krajských měst pro každý zkonstruovaný okruh dle paragrafu 4.1.3 je nižší než 11, pro kontrukci tras použijeme metodu pro malé rozsahy popsanou v paragrafu 3.3.3.2, která pro každý okruh nalezne optimální řešení problému obchodního cestujícího.

4.1.4.1 *Středová varianta rozložení vozového parku*

Sumy vzdáleností a trajektorie vozidel pro jednotlivé okruhy zkonstruované v paragrafu 4.1.3.1:

- 763 km: Praha → Ostrava → Zlín → Praha
- 456 km: Praha → Brno → Hradec Králové → Praha
- 434 km: Praha → Plzeň → Ústí nad Labem → Liberec → Praha
- 840 km: Praha → Olomouc → Jihlava → Karlovy Vary → Praha
- 440 km: Praha → České Budějovice → Pardubice → Praha

Suma vzdáleností pro všechny okruhy je 2933 km.

4.1.4.2 *Krajní varianta rozložení vozového parku*

Sumy vzdáleností a trajektorie vozidel pro jednotlivé okruhy zkonstruované v paragrafu 4.1.3.2:

- 740 km: Praha → Ostrava → Olomouc → Praha
- 599 km: Praha → Zlín → Brno → Praha
- 459 km: Praha → České Budějovice → Jihlava → Pardubice → Praha
- 391 km: Praha → Plzeň → Karlovy Vary → Ústí nad Labem → Praha
- 311 km: Praha → Hradec Králové → Liberec → Praha

Suma vzdáleností pro všechny okruhy je 2500 km.

4.1.5 *Srovnání s ostatními algoritmy*

Problém konstrukce okruhů popsaný 4.1.3 byl řešen Mayerovou metodou (viz 2.3.1) a metodou Fernadez de la Vega-Luecker (viz 2.3.2) s parametrem $r = 0{,}5$.

Vedle Mayerovy metody popsané v paragrafu 2.3.1 uvedeme i výsledky pro modifikaci této metody pro snížení počtu okruhů. Tato varianta při přiřazování krajských měst do okruhu hledá nejbližší krajské město ke krajským městům již přiřazeným do tohoto okruhu. Pokud by kapacita tohoto krajského města překročila kapacitu vozidla, pak vybíráme další nejbližší nepřiřazené krajské město a iterujeme postup, dokud žádné z nepřiřazených měst nelze přidat do okruhu. Takto lze lépe využít kapacitu vozidla a minimalizovat tak počet okruhů. Podrobněji je tato modifikace popsána v [35].

Taktéž pro metodu Fernandez de la Vega-Luecker popsanou v paragrafu 2.3.2 uvedeme modifikaci, která se liší v přiřazování krajských měst, jejichž kapacita je nižší než $\frac{1}{4}$ kapacity vozidla. Varianta popsaná v paragrafu 2.3.2 tato krajská města přiřazuje na základě kapacity, tzn. seřadí tato krajská města dle kapacit sestupně a postupně je přiřazuje do prvního tak, aby nebyla překročena kapacita vozidla tohoto okruhu.

Modifikace metody tato krajská města přiřazuje na základě vzdálenosti. Nepřiřazená krajská města seřadí dle vzdálenosti k nejbližšímu krajskému městu přiřazenému do okruhu. Postupně tato krajská města přiřadí k těmto okruhům tak, aby nebyla překročena kapacita vozidla tohoto okruhu. Podrobněji je tato metoda popsána v [35].

Výsledky pro tyto metody byly převzaty z publikace [35].

Výsledky pro jednotlivé metody a varianty jsou shrnuty v tabulce Tab. 4.2.

Metoda	Cena okruhu (v km)
Navržená metoda dle středového rozložení vozového parku	2933

Navržená metoda dle krajního rozložení vozového parku	2500
Mayerova metoda	2500
Mayerova metoda se sníženým počtem okruhů	2698
Metoda Fernandez de la Vega-Luecker	2548
Metoda Fernandez de la Vega-Luecker s preferováním vzdáleností	2665

Tab. 4.2 Srovnání metod pro řešení problému rozvozu lístků sčítání lidu

Z tabulky Tab. 4.2 je zřejmé, že navržená metoda pro krajní rozložení vozového parku společně s Mayerovou metodou dosahuje nejlepšího výsledku. Propad navržené metodě pro středové rozložení vozového parku lze odůvodnit tím, že vzdálenosti mezi krajskými městy jsou v tomto případě nezanedbatelné a tudíž se neprojeví lepší využití kapacity vozidel. V tomto případě se naopak negativně projeví přiřazování krajských měst pouze na základě kapacit.

Trajektorie konstruované každou ze zmíněných metod jsou obsaženy na přiloženém CD.

4.2 Problém rozvozu potravin

V následujícím paragrafu se zaměříme na problém rozvozu maloobchodního zboží. Jedná se o reálný problém obsloužení většího množství objednávek, přičemž předpokládáme vozový park s vozidly nejednotných parametrů situovaný v místě skladu.

4.2.1 *Vozový park*

Vozový park je umístěn v místě skladu, tj. na adrese Kralický háj 238, PSČ 798 02, Prostějov. Každé vozidlo startuje a končí své okružní jízdy v místě vozového parku. U vozidel známe náklady na kilometr, náklady na hodinu provozu a kapacitu vozidla.

Předpokládáme, že každé vozidlo je omezeno maximální dobou provozu 24 hodin. Do této doby se nezapočítává doba potřebná pro počáteční nakládku zboží, nicméně pokud vozidlo absolvuje více okruhů, pak pro každou nakládku zboží v depu uvažujeme 30 minut.

Seznam dostupných vozidel i s parametry je obsažen na přiloženém CD.

4.2.2 *Objednávky*

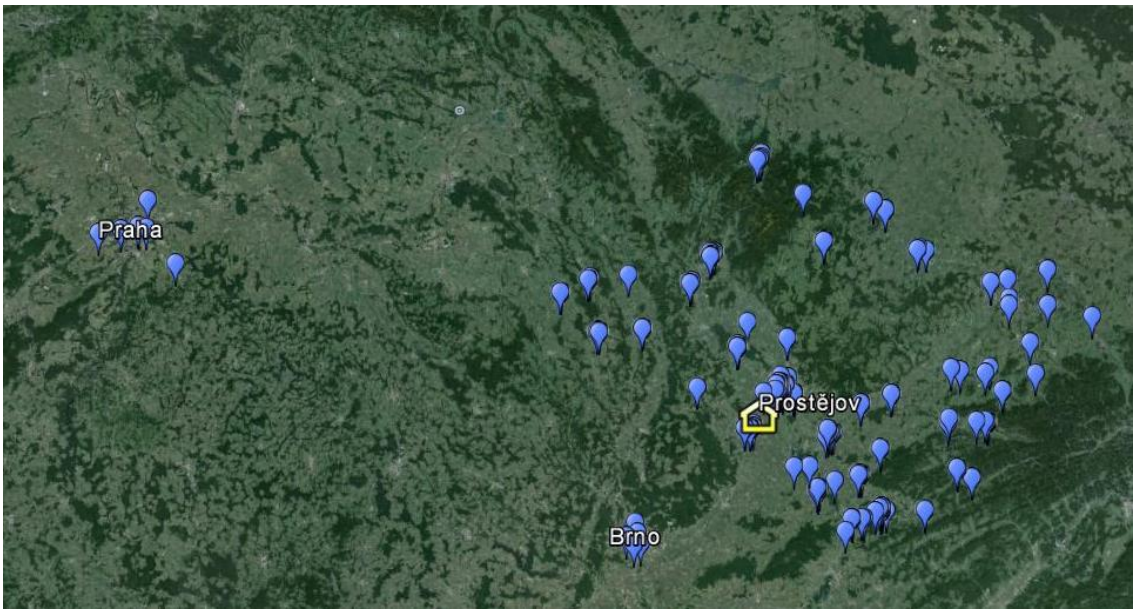
Známe kapacity jednotlivých objednávek a také maximální kapacitu vozidla, která může danou objednávku obsluhovat. V tomto konkrétním případě slouží toto omezení pro případ objednávek v centru měst, které z důvodu nedostupnosti nemohou být obsluhovány velkokapacitními vozidly.

Kapacitu chápejme jako počet palet normované velikosti. Pro obsloužení každé objednávky je stanovena fixní časová dotace 5 minut + 2 minuty za každou vyloženou paletu.

Seznam objednávek i s parametry je obsažen na přiloženém CD.

4.2.3 *Rozložení objednávek*

V následujícím paragrafu se zaměříme na analýzu rozložení objednávek vzhledem k umístění centrálního stanoviště (skladu). Samotnou konstrukci provedeme pomocí mapových podkladů Google Earth.



Obr. 4.2 Rozložení objednávek vůči skladu pro problém rozvozu potravin

Z Obr. 4.2 je zřejmé, že pro rozložení objednávek vzhledem ke skladu nejvíce odpovídá středová varianta popsaná v paragrafu 3.3.2.1.

4.2.4 *Přiřazení objednávek k jednotlivým vozidlům a kontrukce tras*

Při přiřazení objednávek a konstrukci okruhů budeme postupovat dle metody navržené v paragrafu 3.3.2.5.

Dle této metody bylo navrženo 30 okruhů a využito 13 vozidel. Vozidla a objednávky jsou indexovány dle seznamu vozidel a objednávek, který je obsažen na přiloženém CD.

Předpokládejme, že sklad je indexován číslem 340.

Jednotlivé okruhy, celkové časové a distanční nároky jsou shrnuty v následující tabulce

Vozidlo	Okruh	Celková vzdálenost (v km)	Celkový čas (v min)	Náklady na kilometr (v Kč)
13	340->60->57->59->340	60,8	176,4	1,17
	340->81->80->31->340	172,3	314,1	1,17
	340->156->42->58->340	62,8	212,3	1,17
	340->177->24->178->25->340	247,1	448,3	1,17
9	340->88->86->4->45->5->47->94->67->168->165->34->340	352,6	519,3	1,75
	340->49->182->39->38->40->41->157->158->183->133->340	223	428,2	1,75
	340->89->127->12->13->14->15->184->43->48->340	189,8	446,9	1,75
14	340->147->103->102->340	124,3	224,6	1,17
	340->174->87->340	115,1	184,6	1,17
	340->171->131->340	102,1	284	1,17
	340->181->85->84->340	204,7	273,8	1,17
	340->78->21->23->22->340	195,9	400,7	1,17
15	340->162->164->340	145,3	250,1	1,17
	340->50->18->19->340	223,4	417,7	1,17
	340->46->61->155->173->108->340	223,9	364,9	1,17
2	340->96->79->77->98->99->97->54->28->26->27->52->53->340	316,6	518,6	1,75
	340->44->169->163->90->340	301,6	415,1	1,75
1	340->125->189->175->64->119->121->70->340	374,8	538,3	1,75
	340->6->8->7->30->29->33->107->106->105->340	238,5	417,9	1,75
5	340->71->179->93->91->92->128->159->160->161->191->141->340	470,6	659,3	1,75
	340->9->37->109->172->35->82->83->32->110->340	116,4	306,6	1,75
4	340->100->145->3->51->170->17->16->149->340	343,4	560,5	1,75
	340->126->132->185->167->117->115->130->129->72->73->340	380,7	664,3	1,75
6	340->1->113->340	487,7	606,7	1,8
	340->190->180->11->20->154->153->152->176->340	275	380,8	1,8
31	340->68->69->120->122->65->146->114->2->66->340	596,2	1046,1	1,88
22	340->151->111->143->144->186->104->55->56->134->118->116->63->340	950,7	1123,2	1,88
30	340->148->36->10->137->138->136->135->139->140->123->166->75->76->74->340	324,5	566,6	1,88
	340->124->95->150->112->340	332	650,3	1,88
17	340->62->101->187->188->142->340	524	864	1,88

Tab. 4.3 Navržené okruhy a trasy pro problém rozvozu potravin

Pro navržené okruhy je celková vzdálenost 8675,8 km a celkový čas 14264,2 hod. Celkové náklady provozu jsou 14486,28 Kč.

4.2.5 *Srovnání s ostatními algoritmy*

Pro srovnání uvedeme výsledky několika algoritmů, které jsou implementovány v komerčním softwarovém produktu Plantour, který zajišťuje plánování dopravních problémů

velkoobchodních i maloobchodních společností. Tento produkt je v současné době nejvíce využívaným softwarovým řešením pro plánování tras distribučních společností na trhu.

Jedná se o algoritmus Plantour Savings, který je nejvíce používaným obecnějším algoritmem v produktech Plantour a dále nový algoritmus Plantour X1, což je nově implementovaný podpůrný algoritmus plně přizpůsobený konkrétním požadavkům námi testovaného dopravního problému. V následující tabulce jsou uvedeny pouze celkové náklady na provoz. Konkrétní rozložení okruhů včetně tras, celkových vzdáleností a časů je obsaženo na přiloženém CD.

Algoritmus	**Celkové náklady (v Kč)**
Navržená metoda	14486,28
Plantour Savings	-
Plantour X1	7992,4

Tab. 4.4 Srovnání výsledků algoritmů pro problém rozvozu potravin

Z tabulky Tab. 4.4 vidíme, že algoritmus Plantour Savings nedovedl naplánovat trasu vůbec, zatímco algoritmus Plantour X1 dosáhl velice přesvědčivého výsledku v porovnání s námi navrženou metodou.

Důvodem může být fakt, že algoritmus Plantour X1 je plně přizpůsobem konkrétnímu problému rozvozu potravin s uvažovanými speciálními distribučními omezeními, zatímco námi navržený algoritmus postupuje obecněji. Z počtů destinací v každém okruhu je zřejmé, že trasy v jednotlivých okruzích jsou konstruovány většinou optimálně, neboť počet destinací málokdy překračuje číslo 11. Proto výrazné odlišnosti námi navrhovaného algoritmu a algoritmu Plantour X1 jsou především ve fázi rozdělení objednávek k jednotlivým vozidlům. Touto fází bychom se dále chtěli zabývat a uvažovat další distribuční restrikce (jako například podmínky svozu, časová okna, oddělené distribuční prostory vozidel aj.).

Závěr

Metody pro řešení každého reálného dopravního problému jsou svým postupem vždy specifické, neboť pro vybrané případy lze uvažovat takové distribuční restrikce, pro které obecný algoritmus selže. Mnohdy je nutné jednotlivá vozidla k objednávkám přiřadit čistě jen na základě logické úvahy. Je velmi obtížné navrhnout obecný algoritmus, který by dával efektivní výsledky pro různé varianty dopravních problémů, a proto je vhodné každý problém dekomponovat a navrhovat efektivní algoritmy pro jeho vybrané varianty. Volba algoritmu, který bude použit na daný problém, je ponechána na uživateli.

V rámci třetí kapitoly byly navrženy nové algoritmy pro řešení reálných dopravních problémů. Tyto metody byly rozděleny dle rozložení vozových parků k místům obsluhovaných objednávek a dále dle dodatečných podmínek kladených na je obsluhující vozidla.

Metoda určená pro řešení dopravních problémů, kdy místa objednávek nevyžadují splnění speciálních podmínek pro je obsluhující vozidla, byla testována na problému rozvozu lístků pro sčítání lidu z Prahy do krajských měst České republiky. Testovali jsme jak variantu navrhované metody pro centrální rozložení vozového parku ke krajským městům (viz paragraf 3.3.2.1), tak variantu krajního rozložení (viz paragraf 3.3.2.2). Výsledky řešení byly srovnány s výsledky řešení, které daly ostatní metody popsané v rámci druhé kapitoly. Ukázalo se, že varianta nově navrhované metody pro centrální rozložení se zdá být nejvýhodnější v případě „zanedbatelných“ vzdáleností mezi jednotlivými místy objednávek. Naopak varianta řešení v případě krajního rozložení se zdá být efektivní pro „větší“ vzdálenosti mezi jednotlivými místy objednávek.

Metoda určená pro řešení dopravních problémů, kdy místa objednávek vyžadují splnění dalších speciálních podmínek pro je obsluhující vozidla, byla testována na problému distribuce potravin z centrálního skladu v Prostějově do „většího“ počtu míst objednávek v rámci České republiky. Pro tento problém jsme získali reálná data od společnosti, která se zabývá vývojem softwarového řešení pro plánování tras vozidel distribučních společností. Získané výsledky byly porovnány s výsledky řešení od dané společnosti. První algoritmus, který společnost použila, nevedl k žádnému řešení. Druhý algoritmus, který byl specializován

pouze na řešení uvedeného problému, dal uspokojivé výsledky - lepší než výsledky získané námi nově navrženým algoritmem. Důvod může spočívat v tom, že náš algoritmus byl navržen pro řešení obecnějšího problému.

Předložená práce předkládá soubor metod pro řešení obecných dopravních problémů. Je přípravou, která usnadní práci těm, kteří chtějí řešit reálné speciální dopravní problémy. Možným dalším rozšířením navrhovaných algoritmů se zdá být jejich zobecnění v tom směru, že přidáme k podmínkám rozvozu i podmínky svozu zboží, dále podmínky časových oken a oddělených prostorů ve vozidlech při distribuci zboží (suché/chlazené/mražené).

Implementace vybraných (v praxi dříve nejvíce užívaných) algoritmů je obsažena na přiloženém CD. Dále CD obsahuje námi nově navržené algoritmy i s výsledky řešení, které byly testovány na problémech výše uvedených. Přiloženy jsou také seznamy objednávek a vozidel uvažované v těchto dopravních problémech.

Předložená práce je mým prvním pokusem řešit reálné dopravní problémy. O uvedené problematice jsem získal přehled a chtěl bych v tomto směru dále pokračovat.

Seznam použité literatury

1. **Hitchcock, F. L.** The distribution of a product from several sources to numerous localities. *Journal Article.* 1941, stránky 224-230.

2. **Dantzig, G.** Application of the simplex method to a transportation problem. *Activity analysis of production and allocation.* 1951.

3. **Dupačová, J., Gröwe-Kuska, N. a Römisch, W.** Scenario reduction in stochastic programming. *Mathematic Programming.* 2003, stránky 493-511.

4. **Biggs, N. L., Lloyd, E. K. and Wilson, R. J.** *Graph Theory 1736-1936.* Oxford : Clarendon Press, 1986.

5. **Karp, R. M.** *Reducibility, among Combinatorial Problems.* New York : Plennum Press, 1972. pp. 85-104.

6. **Schlag, M., Liao, Y.Z. and Wong, C.K.** An algorithm for optimal two-dimensional compaction of VLSI layouts. *Integration.* 1983, Vol. 1, 2,3.

7. **Toth, P. a Vigo, D.** *The Vehicle Routing Problem.* Philadelphia : SIAM, 2001.

8. **Golden, B. L., Raghavan, S. a Wasil, E. A.** *The Vehicle Routing Problem. Latest Advances and New Challenges.* New York : Springer, 2008.

9. **Miller, C. E., Tucker, A. W. and Zemlin, R. A.** Integer Programming Formulation of Travelling Salesman Problems. *Journal of the ACM.* 1960, Vol. 7, 4, pp. 326-329.

10. **Sniedovich, M.** *Dynamic Programming: Foundation and Principles.* University of Melbourne : Taylor & Francis Group, 2010.

11. **Bellman, Richard.** The theory of dynamic programming. *Bulletin of American Mathematical Society.* 1954, Sv. 60, stránky 503-516.

12. **Clarke, G. a Wright, J.W.** Scheduling of Vehicles from a Central Depot to a Number of Delivery Points. *Operational research.* 1964, stránky 568-581.

13. **Frieze, A. M., Galbiati, G. a Maffioli, F.** On the Worst-Case Performance of some Algorithms for the Assymetric Travelling Salesman Problem. *Networks.* 12, 1982, stránky 23-39.

14. **Kučera, P.** *Metodologie řešení okružního dopravního problému.* Praha : Kučera, 2009.

15. **Kruskal, Joseph B.** On the Shortest Spanning Subtree of a Graph and the Traveling Salesman Problem. *Proceedings of the American Mathematical Society.* 1956, Sv. 7, 1, stránky 48-50.

16. **Borůvka, Otakar.** *O jistém problému minimálním.* Brno : Mor. přírodovědecká společnost, 1926.

17. **Jarník, Vojtěch.** O jistém problému minimálním. *Práce moravské přírodovědecké společnosti.* 1930, Sv. 6, stránky 57-63.

18. **Chazelle, Bernard.** A Minimum Spanning Tree Algorithm with Inverse-Ackermann Type Complexity. *Journal of the ACM.* 2000, Sv. 47, 6, stránky 1028-1047.

19. **Kolář, Josef.** *Teoretická informatika.* Praha : Česká informatická společnost, 2004.

20. *Worst-case analysis of a new heuristic for the travelling salesman problem.* **Christofides, Nicos.** CMU : Graduate School of Industrial Administration, 1976.

21. **Pelikán, Jan.** *Diskrétní modely v operačním výzkumu.* Praha : Professional Publishing, 2001.

22. **Rosenkrantz, Daniel J., Stearns, Richard E. a Lewis, Philip M.** An Analysis of Several Heuristics for the Travelling Salesman Problem. *SIAM Journal on Computing.* 1977, Sv. 6, 3, stránky 563-581.

23. **Grygarová, Libuše.** *Úvod do lineárního programování.* Praha : Státní pedagogické nakladatelství, 1975.

24. **Grassé, P. P.** La reconstruction du nid et les coordinations interindividuelles chez Belicositermes natalensis et Cubitermes sp. La théorie de la Stigmergie : Essai d'interprétation du comportement des termites constructeurs. *Insectes Sociaux.* 6, 1959.

25. **Dorigo, M. a Gambardella, L. M.** Ant Colony System : A Cooperative Learning Approach to the Traveling Salesman Problem. *IEEE Transactions on Evolutionary Computation.* 1997, Sv. 1, 1, stránky 53-66.

26. **Dorigo, M., Caro, G. D. a Sampels, M.** *Ant Algorithms: Third International Workshop.* Brussel : Springer, 2002.

27. **Dorigo, M.** *Optimization, Learning and Natural Algorithms.* Milano : Politecnico de Milano, 1992.

28. **Dorigo, M. a Stützle, T.** *Ant colony optimization.* Cambridge : MIT Press, 2004.

29. **Karp, Richard M.** A patching algorithm for the non-symmetric traveling-salesman problem. *Society fot Industrial and Applied Mathematics.* 1979, Sv. 8, 4.

30. *The Probabilistic Relationship between the Assignment Problem and Assymetric Travelling Salesman Problem .* **Frieze, A. M. a Sorkin, G. B.** místo neznámé : 12. Annual Symposium on Discrete Algorithms (SODA), 2001.

31. *Construction Heuristics and Domination Analysis for the Assymetric TSP.* **Glover, F., a další.** místo neznámé : European Journal of Operational Research, 2001.

32. *Iterative Patching and the Assymetric Traveling Salesman Problem.* **Turkensteena, M., a další.** University of Groningen : Discrete Optimalization, 2006.

33. **Kosková, Ivanka Ing.** *Distribuční úlohy I.* Praha : Česká zemědělská univerzita v Praze, 2006.

34. **Habr, Jaroslav.** *Jednoduché metody pro ekonomickou praxi.* Praha : Řada ekonomické literatury, 1966.

35. **Kuhn, Harold W.** The Hungarian Method for the assignement problem. *Naval Research Logistics Quarterly.* 2, 1955.

36. **Vega, Fernandez de la a Luecker, G. S.** Bin packing can be solved within 1+e in linear time. *Combinatorica.* 1, 1981, stránky 349-355.

37. *Využití metod pro řešení "bin packing" problému pro okružní dopravní problém.* **Dömeová, L. a Kučera, P.** Praha : Česká zemědělská univerzita v Praze, 2006.

Seznam tabulek

Seznam obrázků